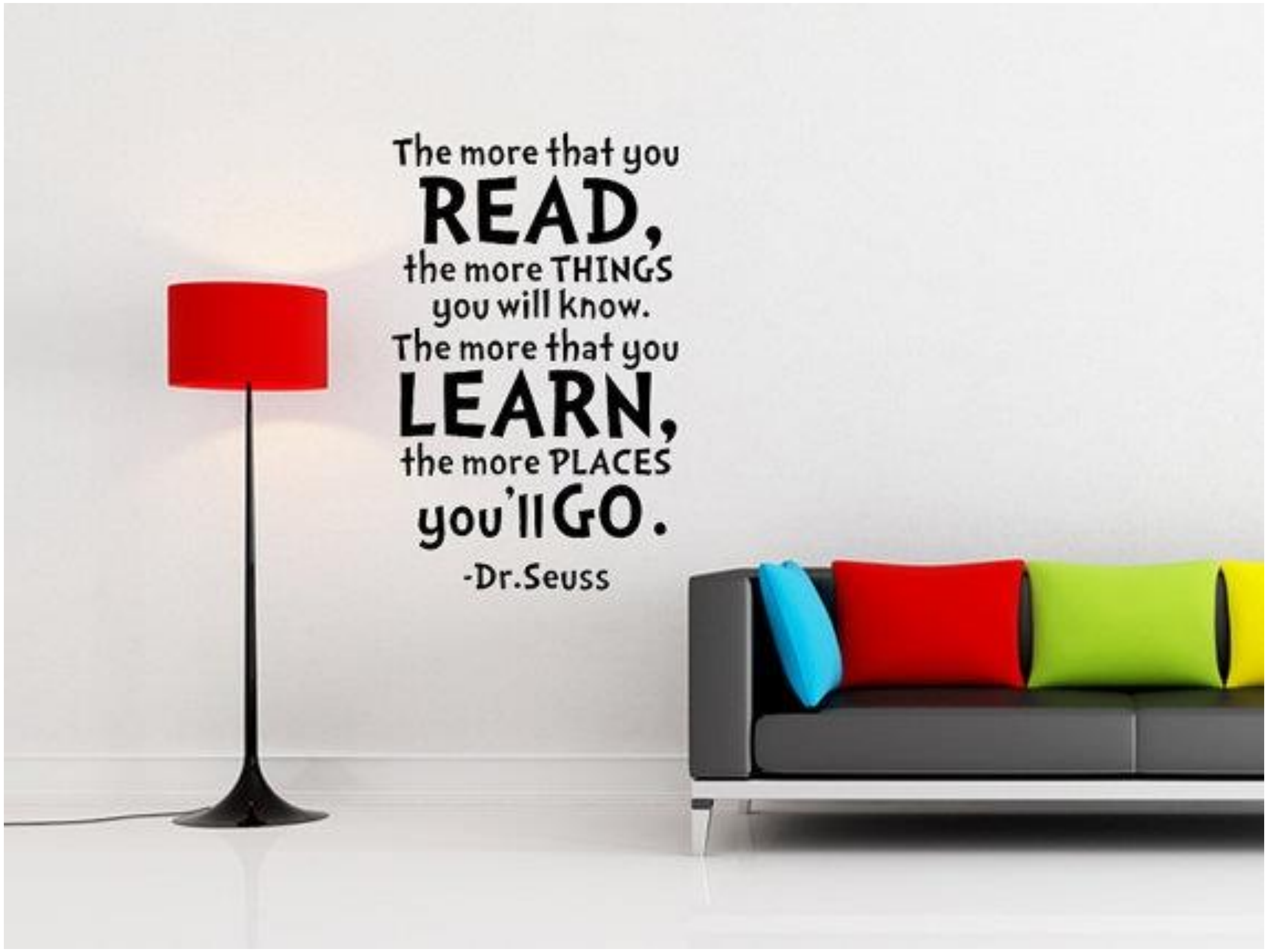
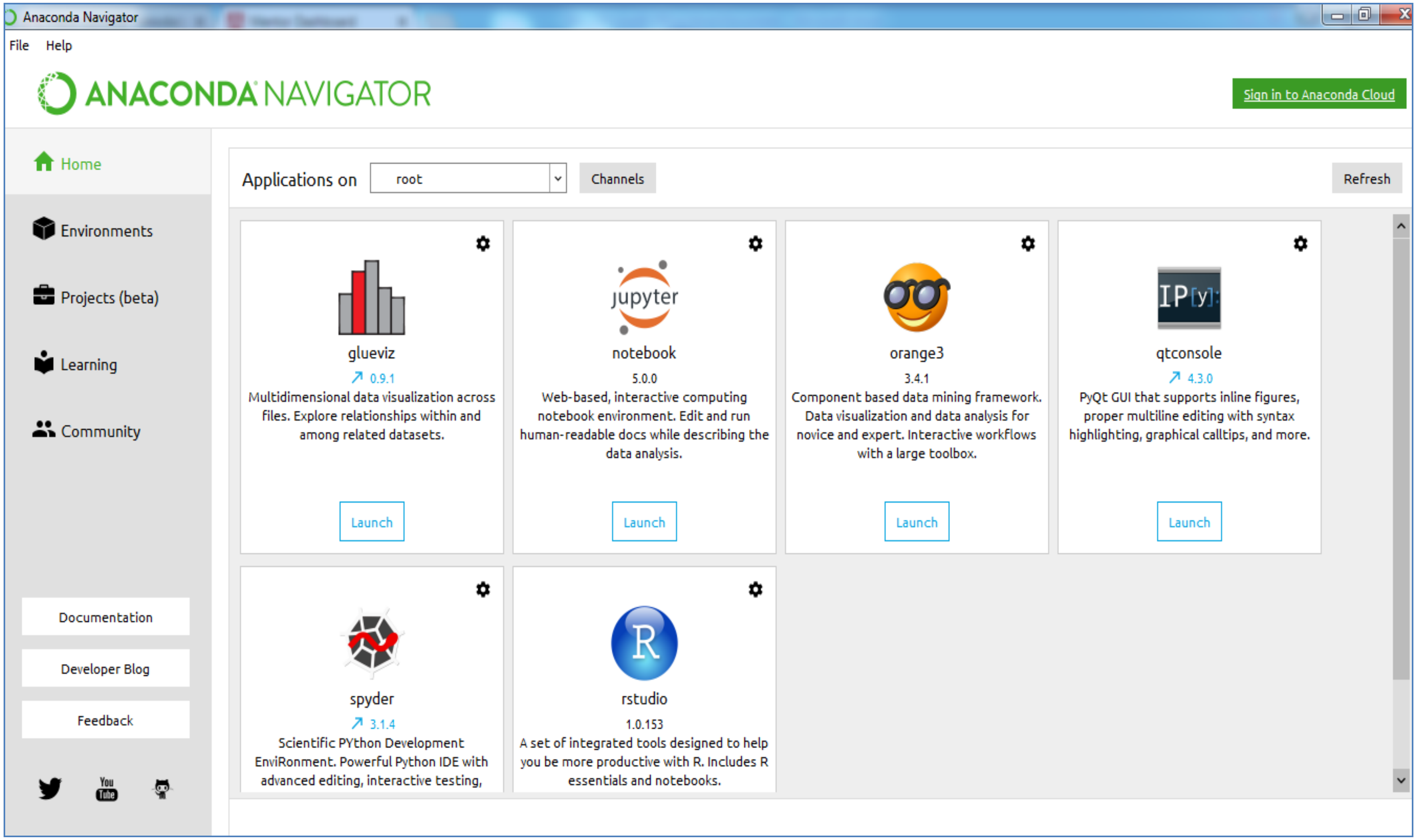
Python


The more that you
READ,
the more THINGS
you will know.
The more that you
LEARN,
the more PLACES
you'll GO.
-Dr.Seuss

Anaconda Navigator



Applications on root

glueviz
0.9.1
Multidimensional data visualization across files. Explore relationships within and among related datasets.

notebook
5.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

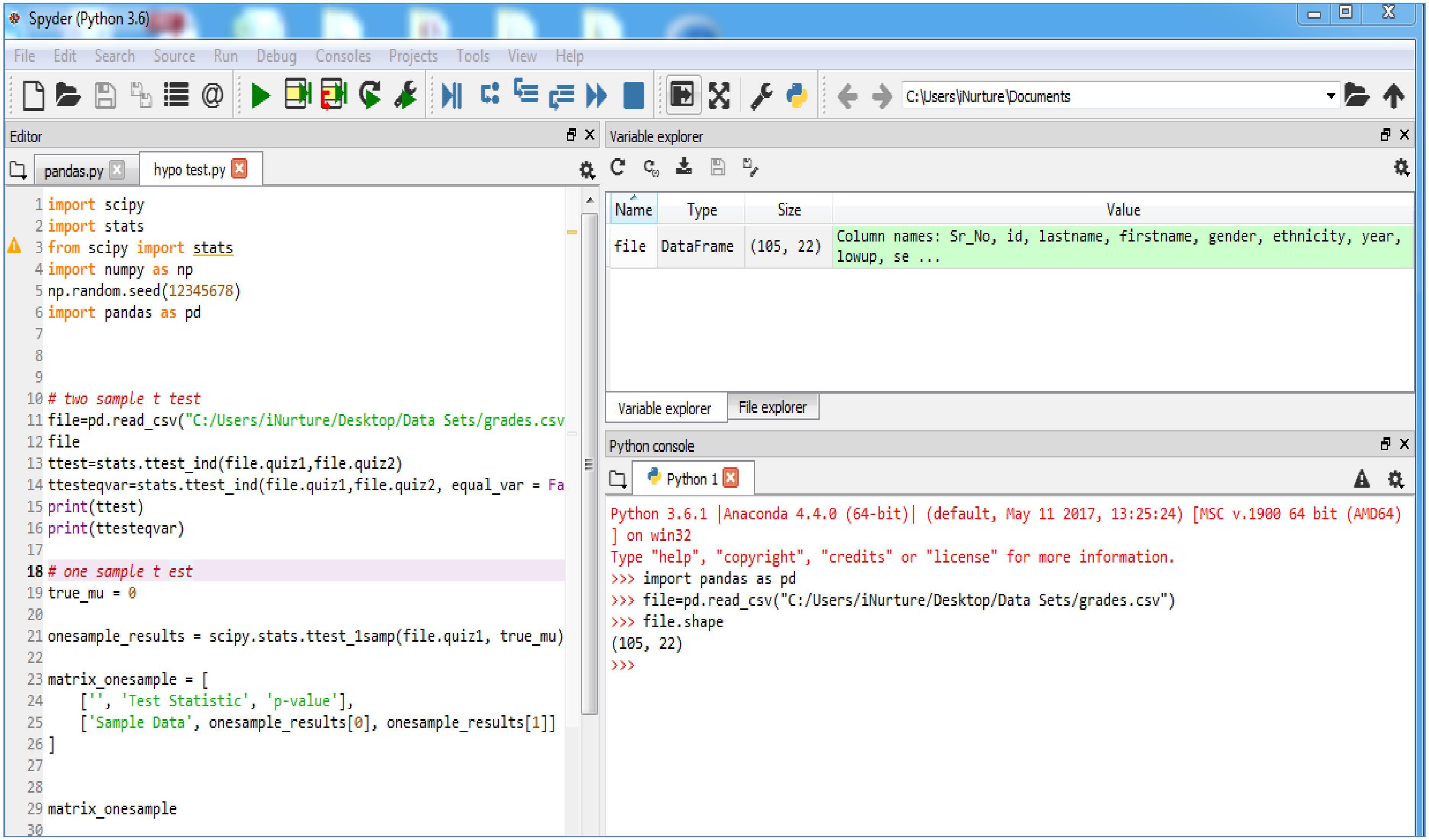
orange3
3.4.1
Component based data mining Framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

qtconsole
4.3.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

spyder
3.1.4
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing,

rstudio
1.0.153
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Spyder

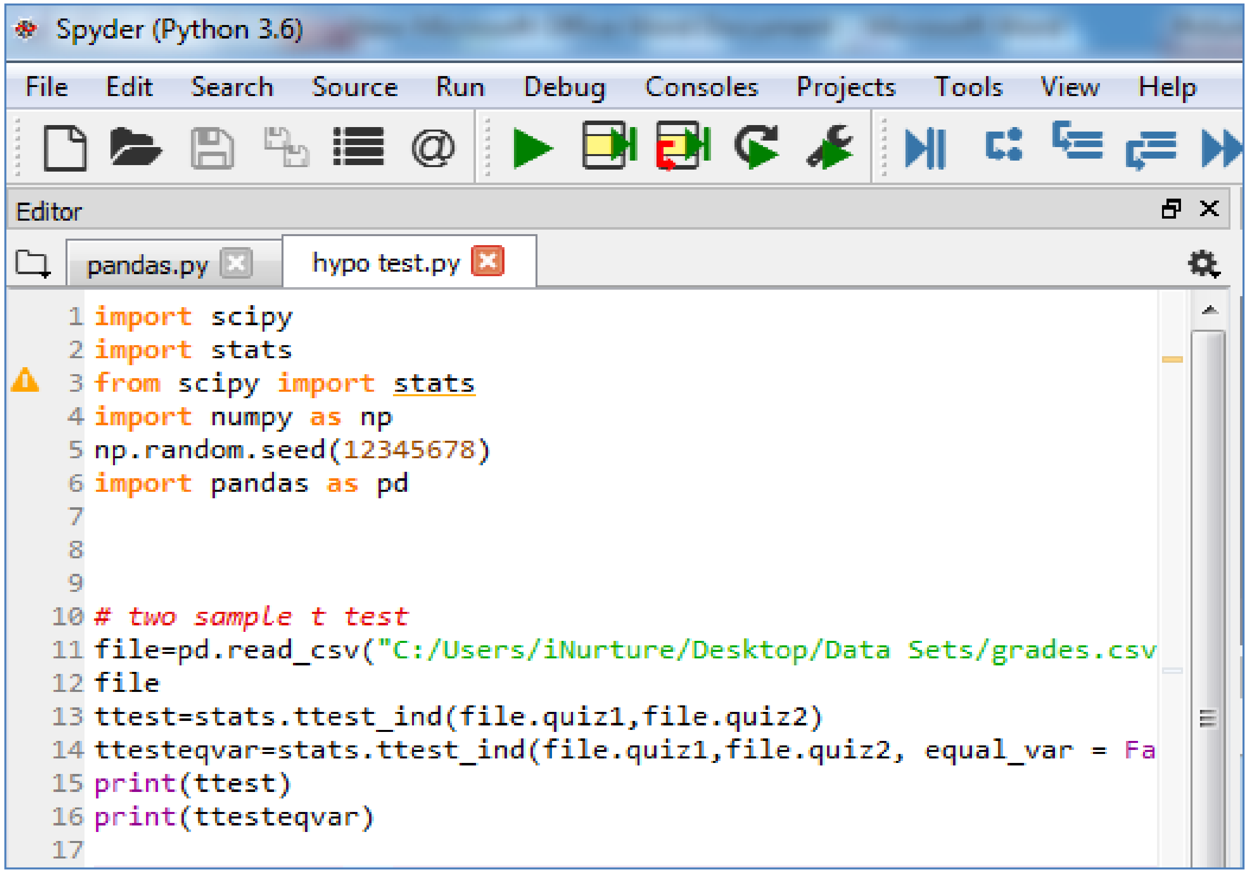


```python
import scipy
import stats
from scipy import stats
import numpy as np
np.random.seed(12345678)
import pandas as pd



# two sample t test
file=pd.read_csv("C:/Users/iNurture/Desktop/Data Sets/grades.csv
file
ttest=stats.ttest_ind(file.quiz1,file.quiz2)
ttesteqvar=stats.ttest_ind(file.quiz1,file.quiz2, equal_var = Fa
print(ttest)
print(ttesteqvar)

# one sample t est
true_mu = 0

onesample_results = scipy.stats.ttest_1samp(file.quiz1, true_mu)

matrix_onesample = [
    ['', 'Test Statistic', 'p-value'],
    ['Sample Data', onesample_results[0], onesample_results[1]]
]


matrix_onesample
```

Name	Type	Size	Value
file	DataFrame	(105, 22)	Column names: Sr_No, id, lastname, firstname, gender, ethnicity, year, lowup, se ...

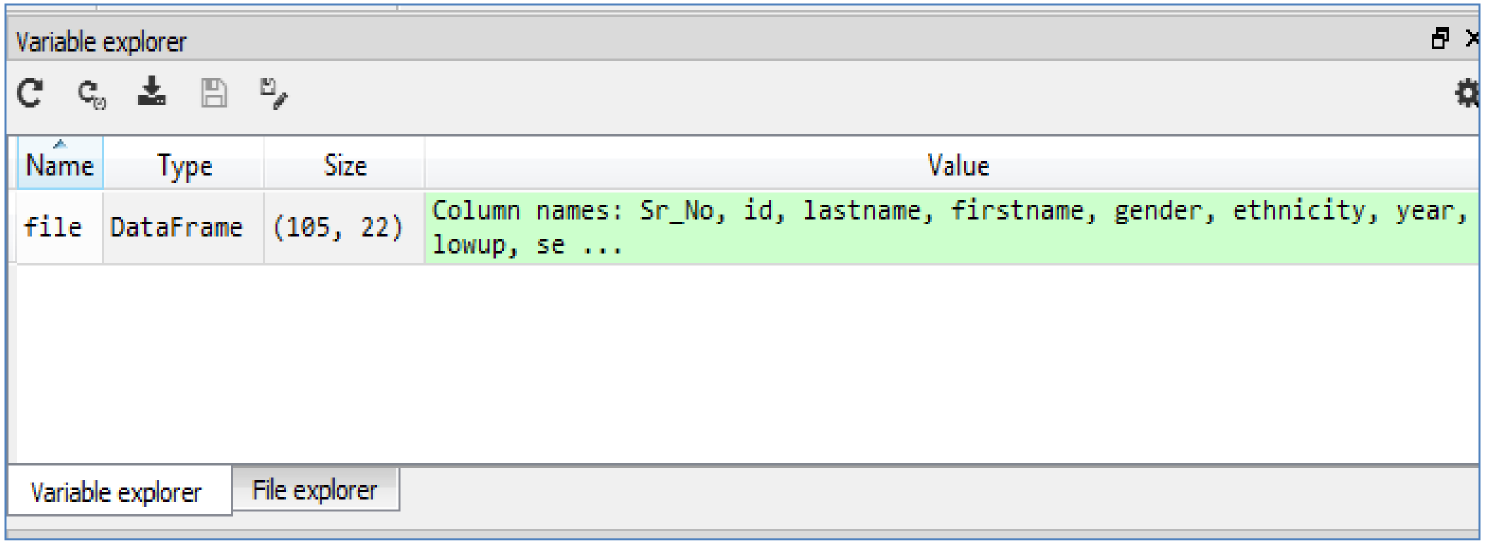
```
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit (AMD64)
] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> file=pd.read_csv("C:/Users/iNurture/Desktop/Data Sets/grades.csv")
>>> file.shape
(105, 22)
>>>
```

Script


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor
pandas.py
hypo test.py

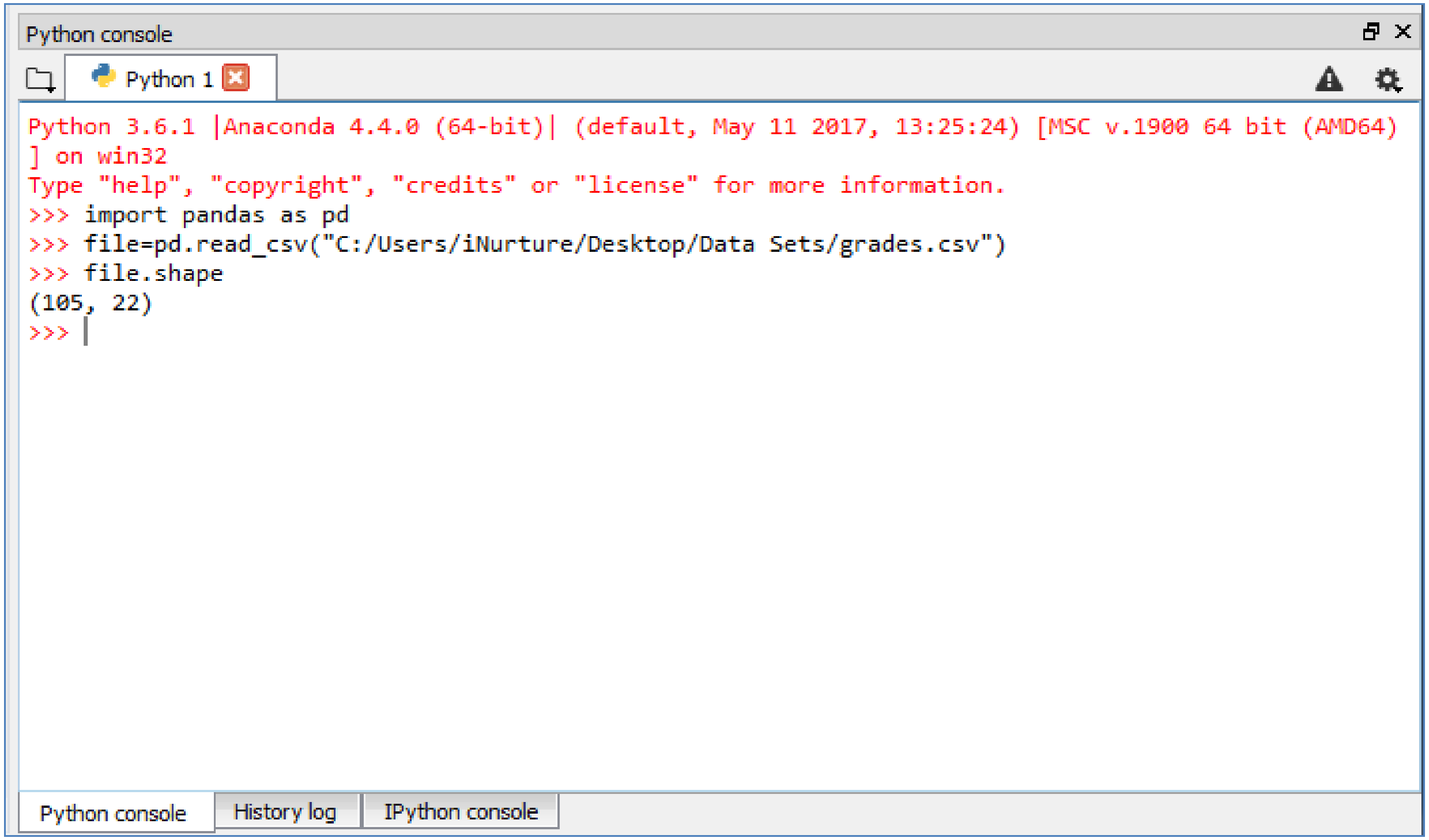
```
import scipy
import stats
from scipy import stats
import numpy as np
np.random.seed(12345678)
import pandas as pd



# two sample t test
file=pd.read_csv("C:/Users/iNurture/Desktop/Data Sets/grades.csv
file
ttest=stats.ttest_ind(file.quiz1,file.quiz2)
ttesteqvar=stats.ttest_ind(file.quiz1,file.quiz2, equal_var = Fa
print(ttest)
print(ttesteqvar)

```

R like Global Environment



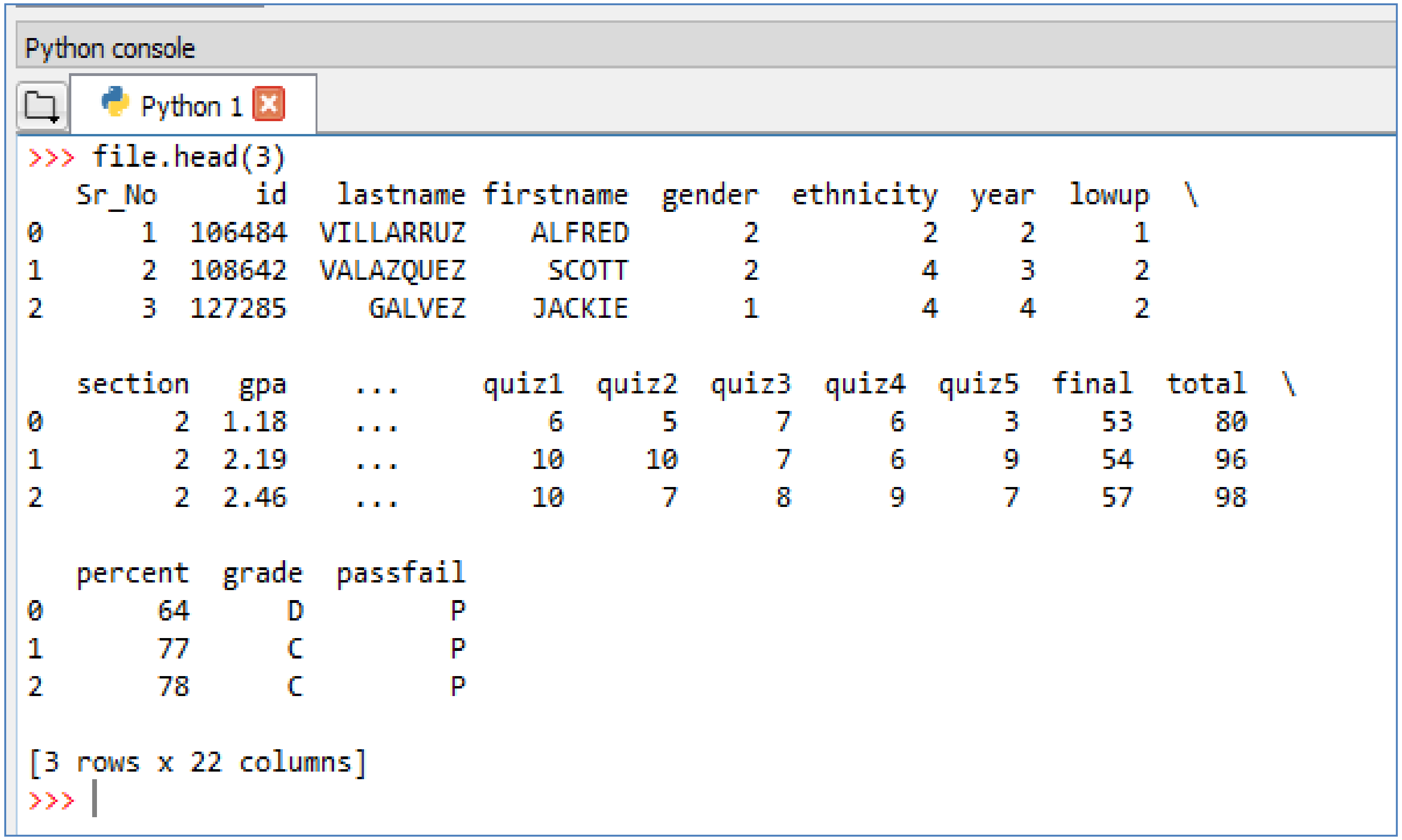
Name	Type	Size	Value
file	DataFrame	(105, 22)	Column names: Sr_No, id, lastname, firstname, gender, ethnicity, year, lowup, se ...

R like console



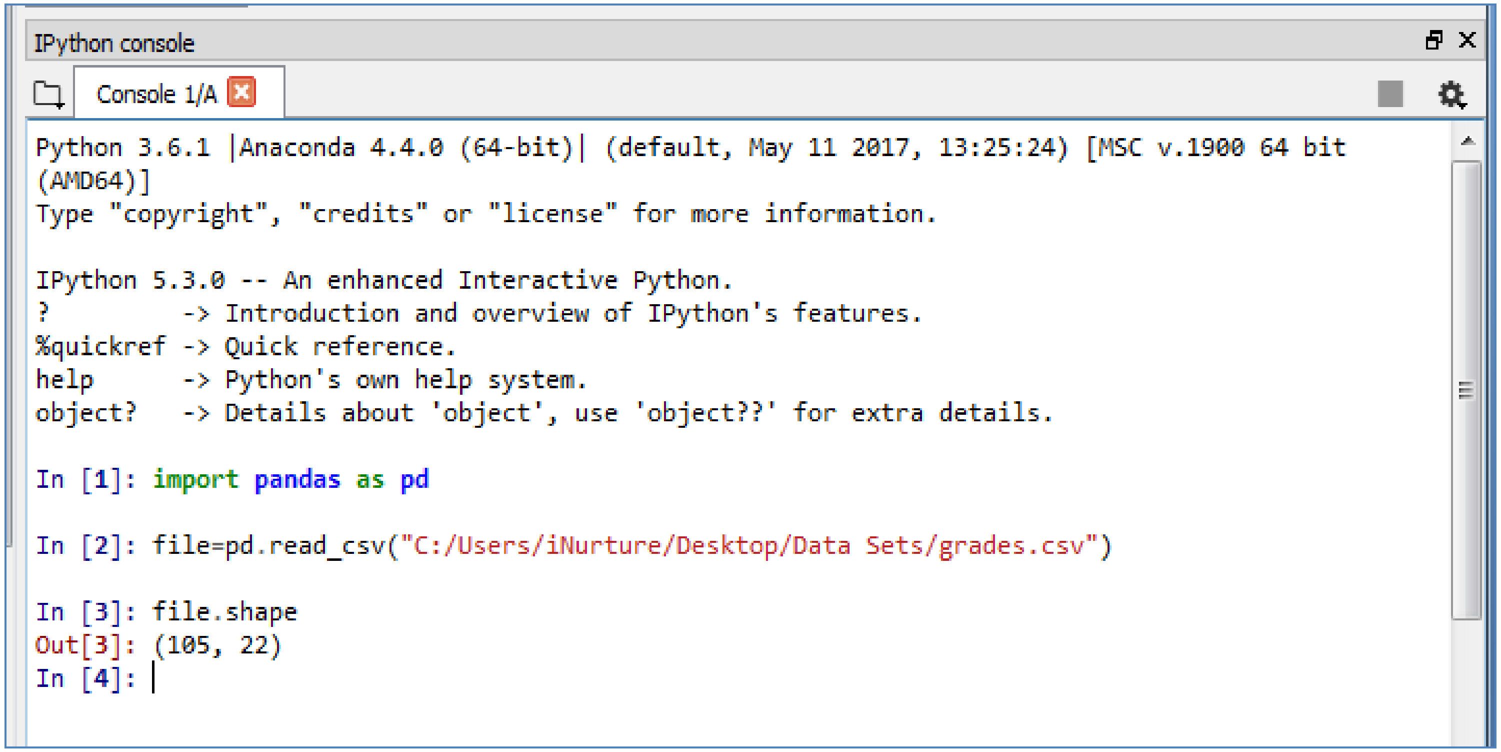
```
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit (AMD64)
] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> file=pd.read_csv("C:/Users/iNurture/Desktop/Data Sets/grades.csv")
>>> file.shape
(105, 22)
>>> 
```

R like head(3)



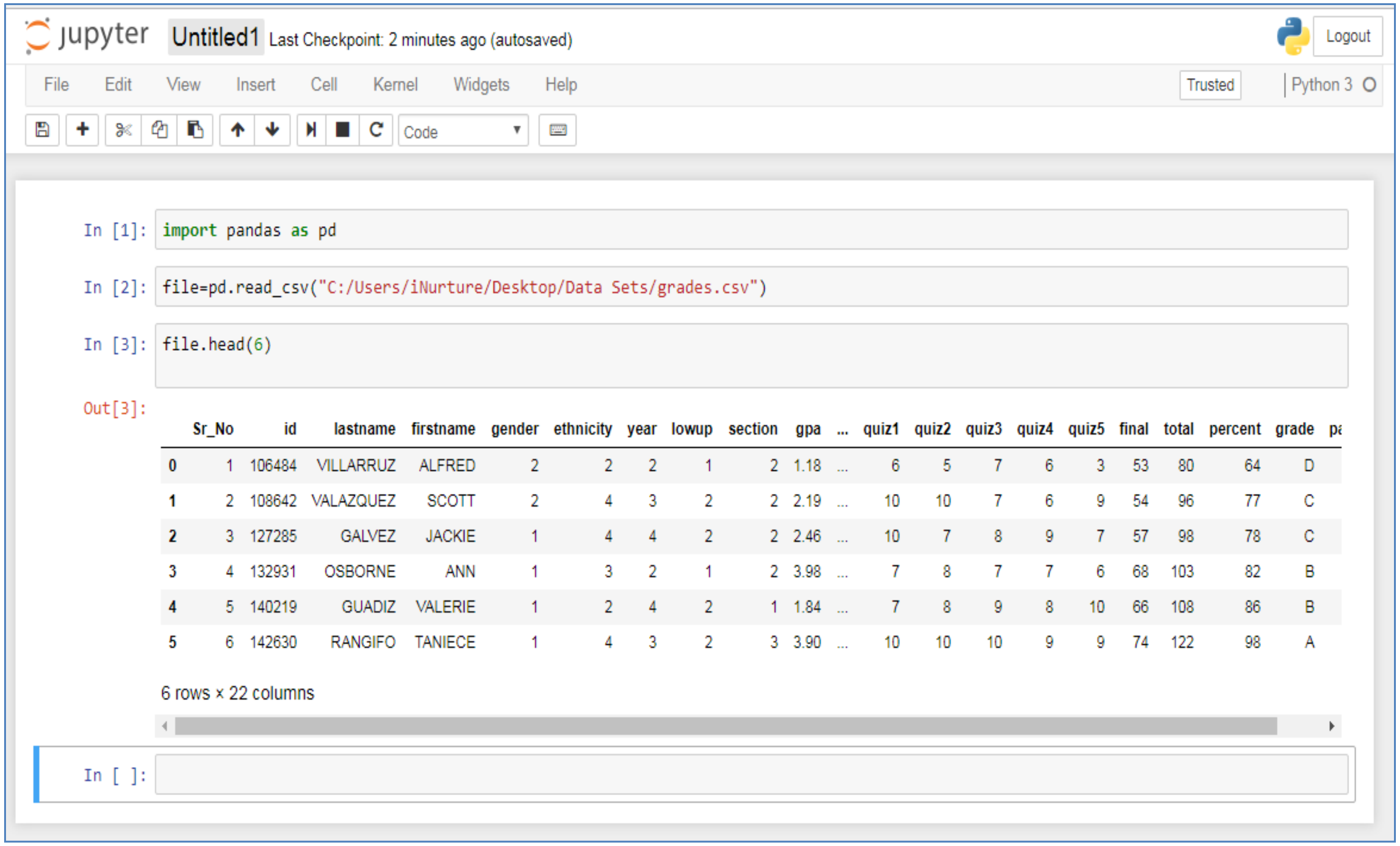
```
>>> file.head(3)
   Sr_No      id   lastname firstname  gender  ethnicity  year  lowup  \
0      1  106484  VILLARRUZ    ALFRED       2          2     2      1
1      2  108642  VALAZQUEZ     SCOTT       2          4     3      2
2      3  127285     GALVEZ    JACKIE       1          4     4      2

   section   gpa   ...    quiz1  quiz2  quiz3  quiz4  quiz5  final  total  \
0        2  1.18   ...        6      5      7      6      3     53     80
1        2  2.19   ...       10     10      7      6      9     54     96
2        2  2.46   ...       10      7      8      9      7     57     98

   percent  grade  passfail
0       64      D         P
1       77      C         P
2       78      C         P

[3 rows x 22 columns]
>>> 
```

IPython console



```
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import pandas as pd

In [2]: file=pd.read_csv("C:/Users/iNurture/Desktop/Data Sets/grades.csv")

In [3]: file.shape
Out[3]: (105, 22)
In [4]:
```

jupyter



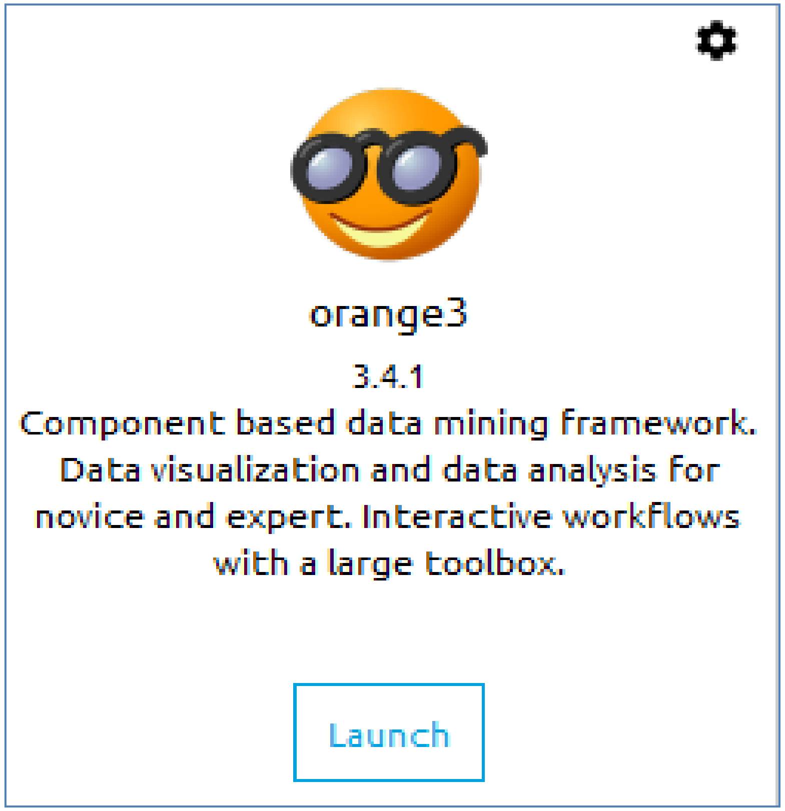
```
In [1]: import pandas as pd

In [2]: file=pd.read_csv("C:/Users/iNurture/Desktop/Data Sets/grades.csv")

In [3]: file.head(6)

Out[3]:
```

	Sr_No	id	lastname	firstname	gender	ethnicity	year	lowup	section	gpa	...	quiz1	quiz2	quiz3	quiz4	quiz5	final	total	percent	grade	pa
0	1	106484	VILLARRUZ	ALFRED	2	2	2	1	2	1.18	...	6	5	7	6	3	53	80	64	D	
1	2	108642	VALAZQUEZ	SCOTT	2	4	3	2	2	2.19	...	10	10	7	6	9	54	96	77	C	
2	3	127285	GALVEZ	JACKIE	1	4	4	2	2	2.46	...	10	7	8	9	7	57	98	78	C	
3	4	132931	OSBORNE	ANN	1	3	2	1	2	3.98	...	7	8	7	7	6	68	103	82	B	
4	5	140219	GUADIZ	VALERIE	1	2	4	2	1	1.84	...	7	8	9	8	10	66	108	86	B	
5	6	142630	RANGIFO	TANIECE	1	4	3	2	3	3.90	...	10	10	10	9	9	74	122	98	A	

6 rows × 22 columns

```
In [ ]:
```

orange: Data Mining Tool

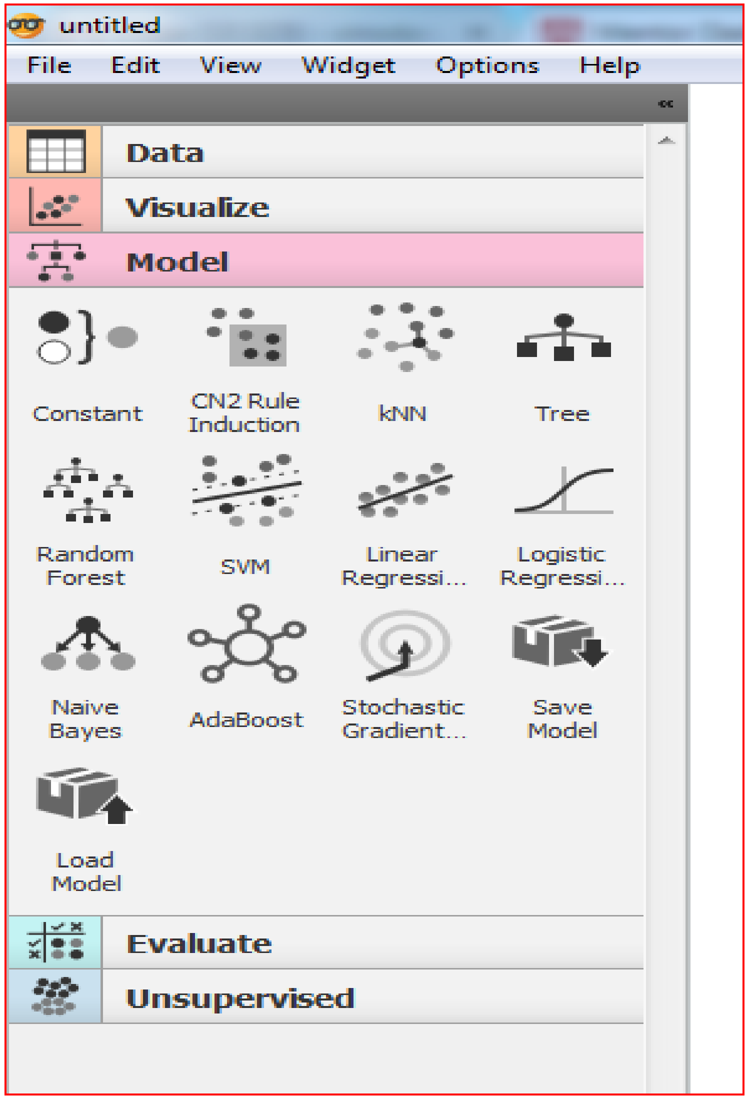


orange3

3.4.1

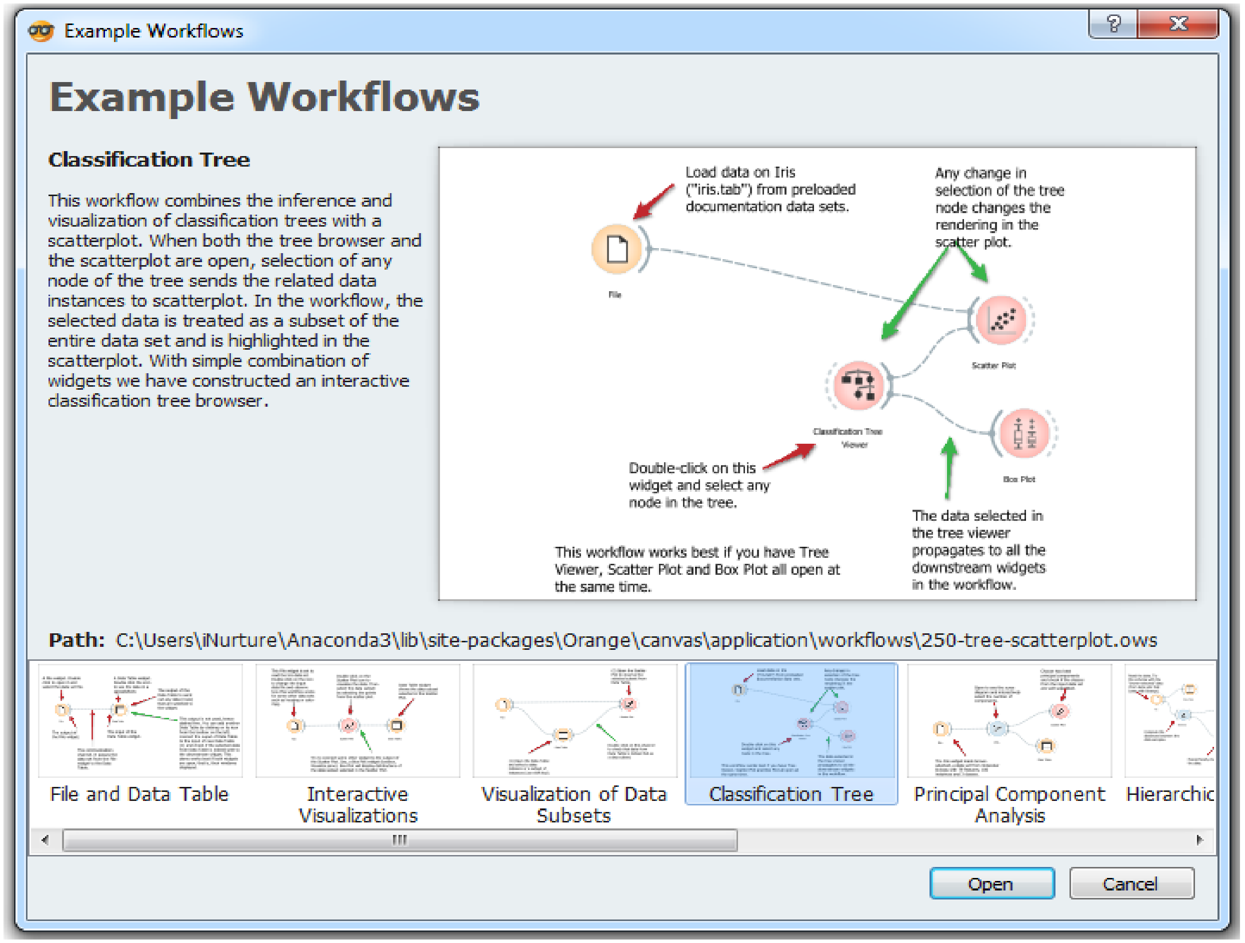
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Launch



Go to help → Example

Example Classification Tree



Example Workflows

Example Workflows

Classification Tree

This workflow combines the inference and visualization of classification trees with a scatterplot. When both the tree browser and the scatterplot are open, selection of any node of the tree sends the related data instances to scatterplot. In the workflow, the selected data is treated as a subset of the entire data set and is highlighted in the scatterplot. With simple combination of widgets we have constructed an interactive classification tree browser.

Load data on Iris ("iris.tab") from preloaded documentation data sets.

Any change in selection of the tree node changes the rendering in the scatter plot.

Double-click on this widget and select any node in the tree.

The data selected in the tree viewer propagates to all the downstream widgets in the workflow.

This workflow works best if you have Tree Viewer, Scatter Plot and Box Plot all open at the same time.

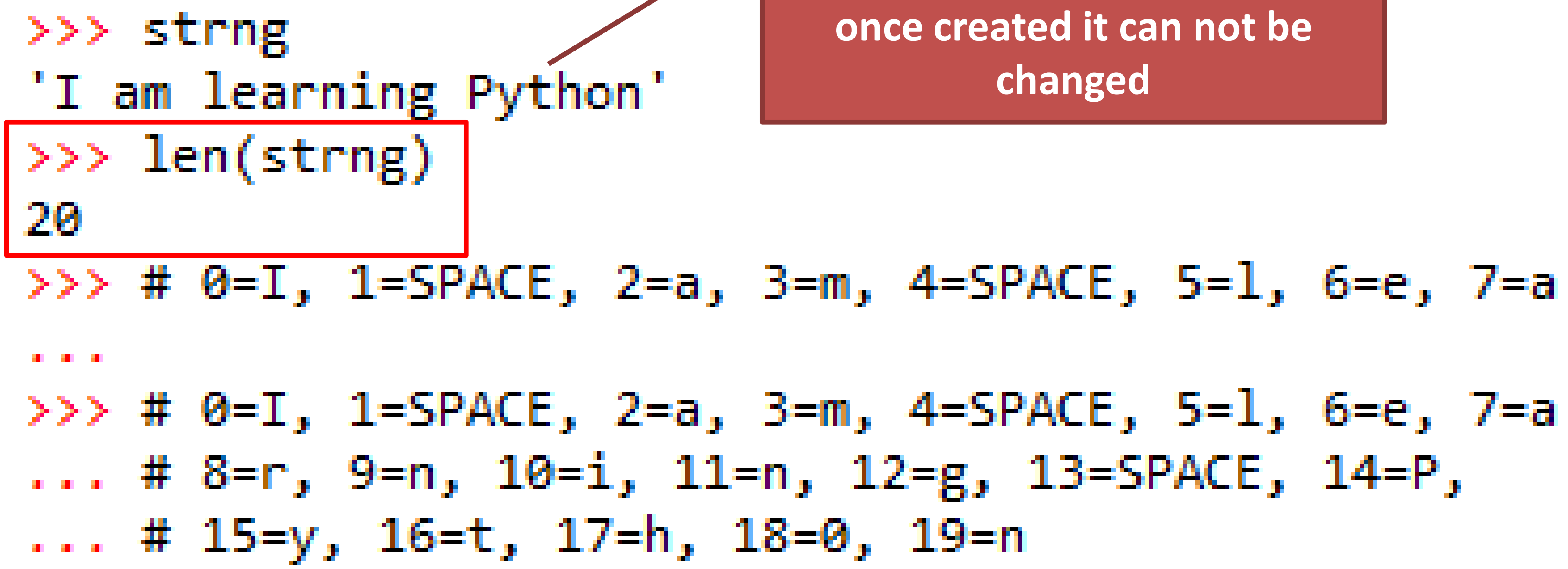
Path: C:\Users\iNurture\Anaconda3\lib\site-packages\Orange\canvas\application\workflows\250-tree-scatterplot.ows

File
Scatter Plot
Classification Tree Viewer
Box Plot
File and Data Table
Interactive Visualizations
Visualization of Data Subsets
Classification Tree
Principal Component Analysis
Hierarchic
Open
Cancel

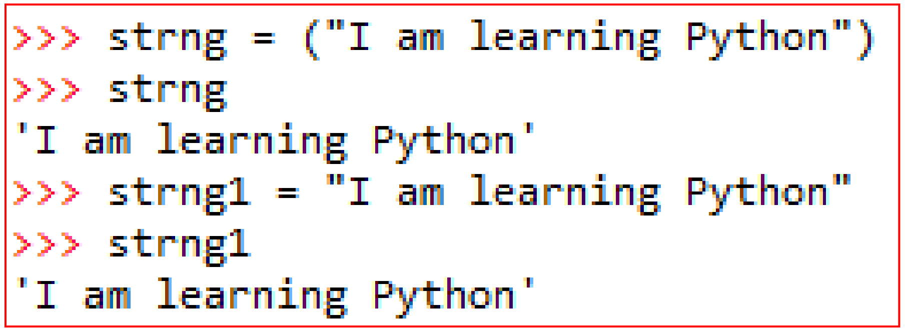
Your teacher for orange

String

String is immutable means once created it can not be changed

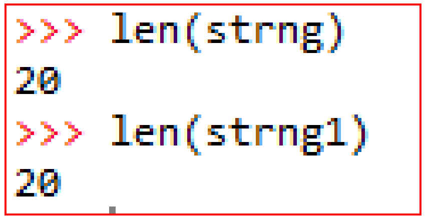


```
>>> strng
'I am learning Python'
>>> len(strng)
20
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
...
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
... # 8=r, 9=n, 10=i, 11=n, 12=g, 13=SPACE, 14=P,
... # 15=y, 16=t, 17=h, 18=0, 19=n
```



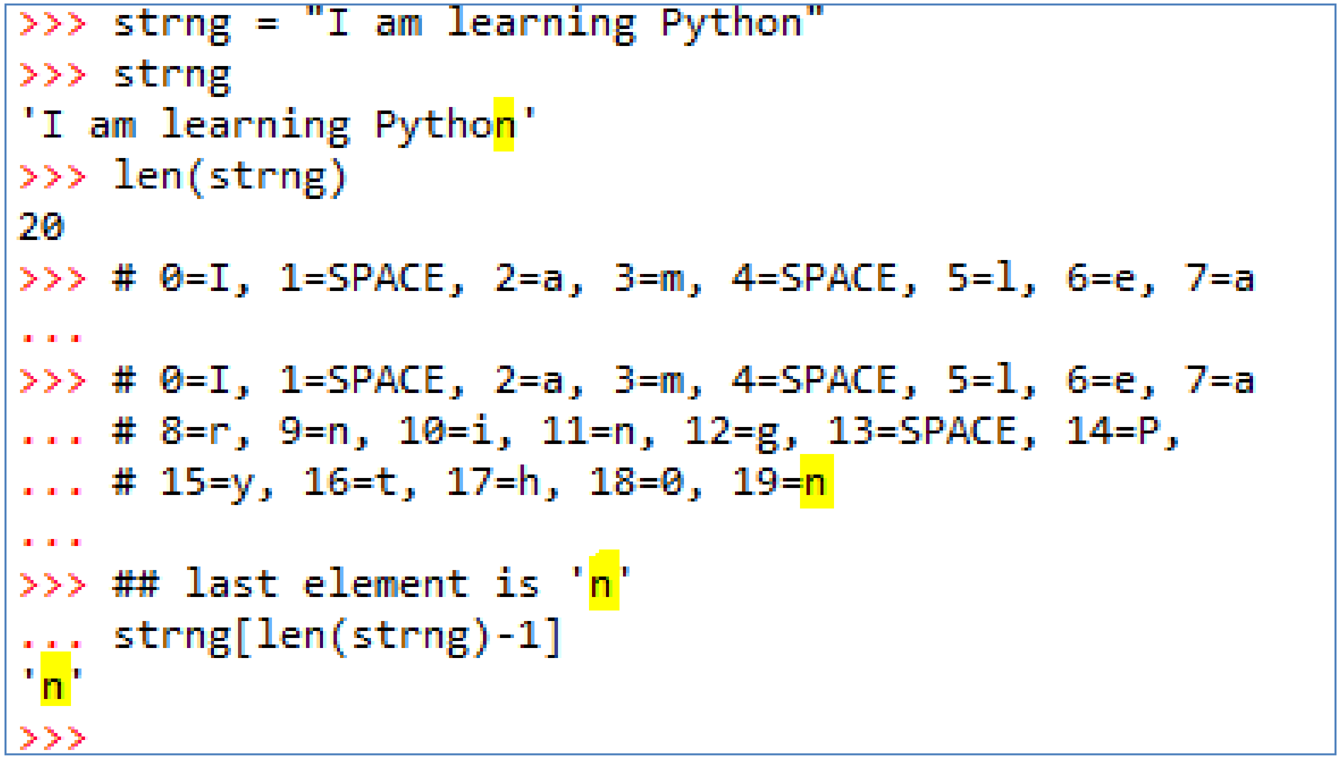
```
>>> strng = ("I am learning Python")
>>> strng
'I am learning Python'
>>> strng1 = "I am learning Python"
>>> strng1
'I am learning Python'
```

Parenthesis is immaterial

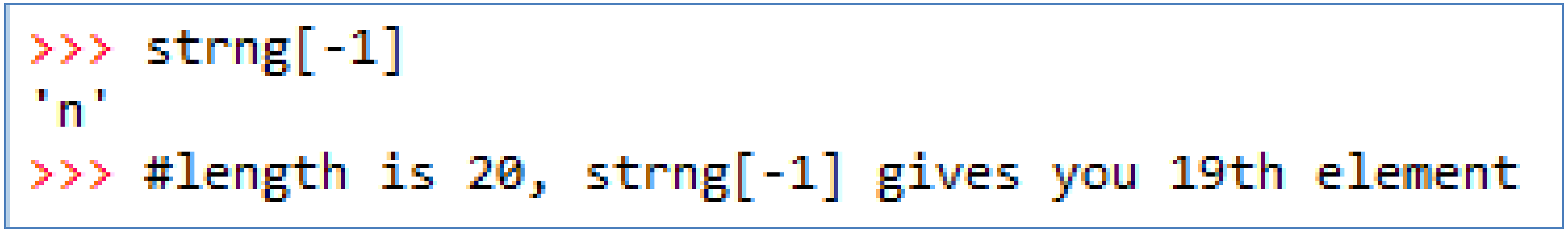


```
>>> len(strng)
20
>>> len(strng1)
20
```

String

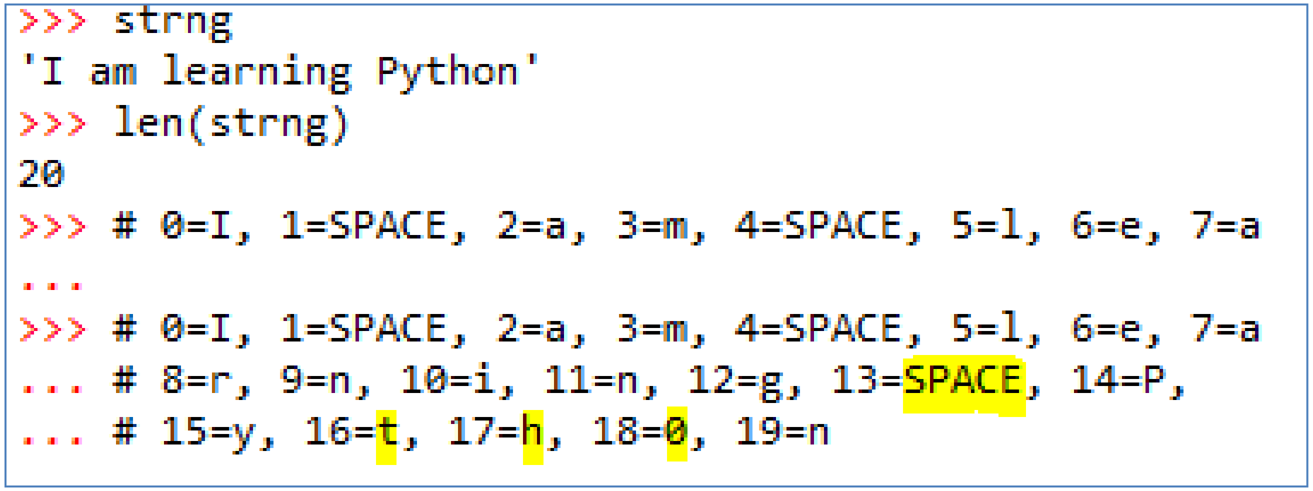


```
>>> strng = "I am learning Python"
>>> strng
'I am learning Python'
>>> len(strng)
20
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
...
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
... # 8=r, 9=n, 10=i, 11=n, 12=g, 13=SPACE, 14=P,
... # 15=y, 16=t, 17=h, 18=0, 19=n
...
>>> ## last element is 'n'
... strng[len(strng)-1]
'n'
>>>
```



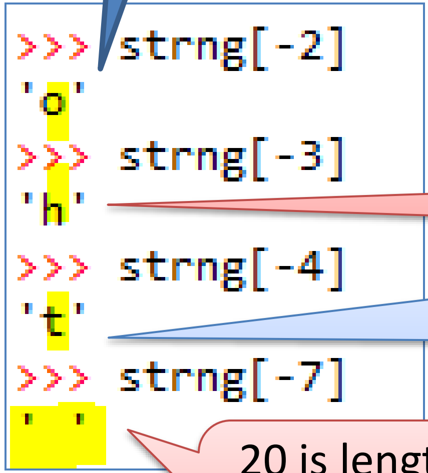
```
>>> strng[-1]
'n'
>>> #length is 20, strng[-1] gives you 19th element
```

String



```
>>> strng
'I am learning Python'
>>> len(strng)
20
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
...
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
... # 8=r, 9=n, 10=i, 11=n, 12=g, 13=SPACE, 14=P,
... # 15=y, 16=t, 17=h, 18=0, 19=n
```

20 is length
20-2=18 is
‘o’

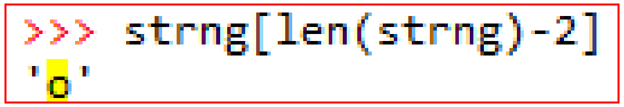


```
>>> strng[-2]
'o'
>>> strng[-3]
'h'
>>> strng[-4]
't'
>>> strng[-7]
' '
```

20 is length
20-3=17 is
‘h’

Same as
strng[-2]

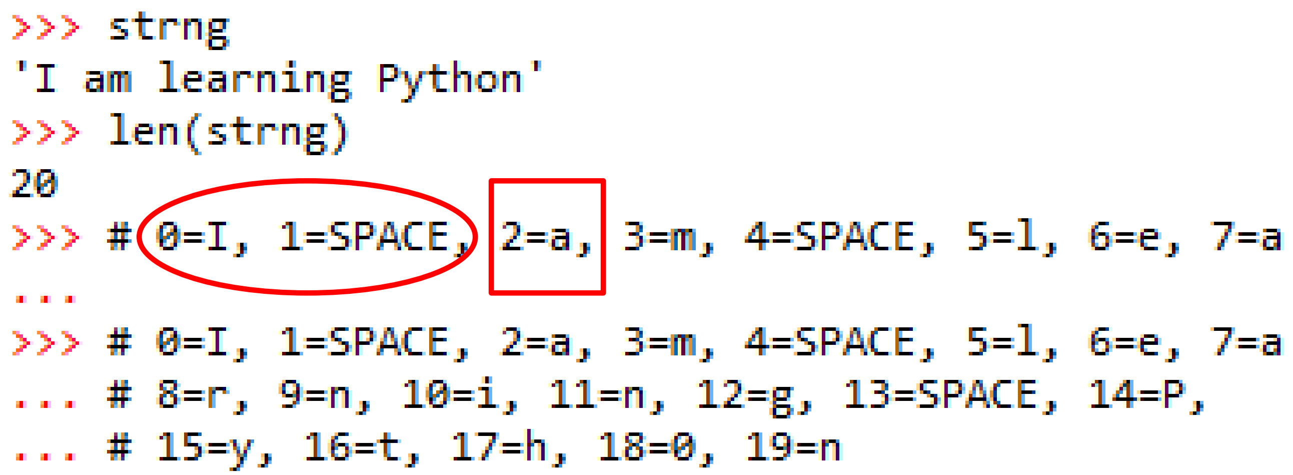
20 is length
20-2=16 is
‘t’



```
>>> strng[len(strng)-2]
'o'
```

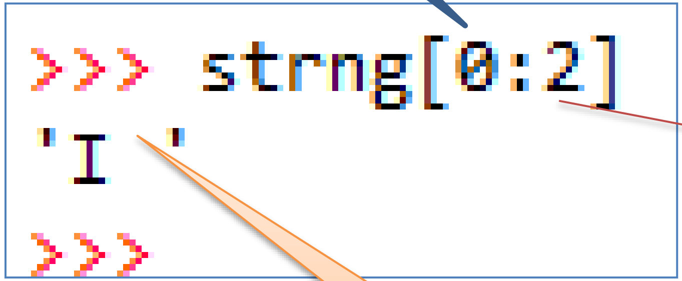
20 is length
20-7=13 is
‘ ’ *space*

String



```
>>> strng
'I am learning Python'
>>> len(strng)
20
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
...
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
... # 8=r, 9=n, 10=i, 11=n, 12=g, 13=SPACE, 14=P,
... # 15=y, 16=t, 17=h, 18=0, 19=n
```

Start from 0 which is I



```
>>> strng[0:2]
'I '
>>>
```

Last number 2, which is 'a' WILL NOT BE INCLUDED

Goes to 1 which is ***space***

String

```
>>> strng
'I am learning Python'
>>> len(strng)
20
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
...
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
... # 8=r, 9=n, 10=i, 11=n, 12=g, 13=SPACE, 14=P,
... # 15=y, 16=t, 17=h, 18=0, 19=n
```

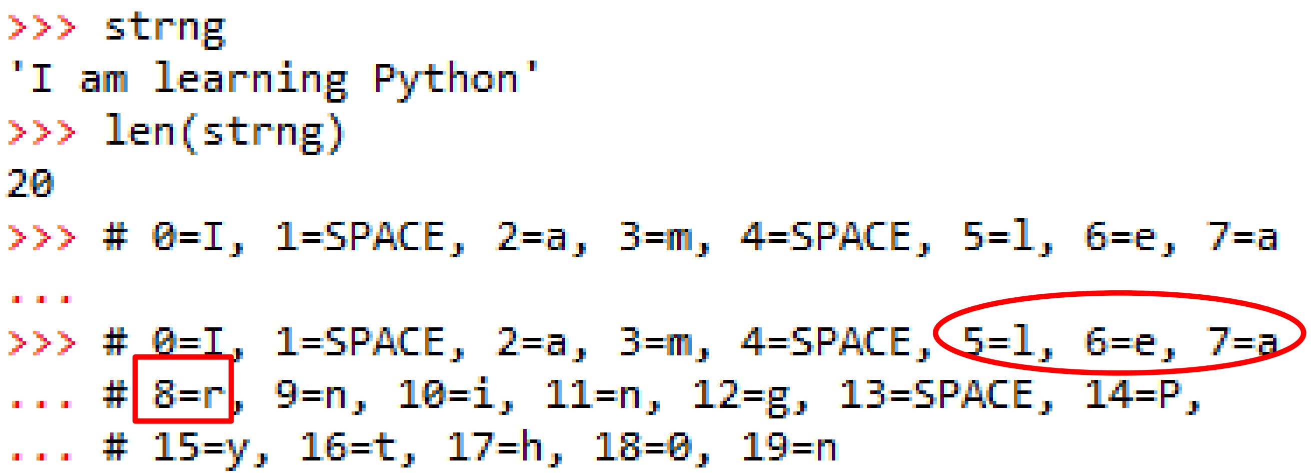
Start from 3 which is m

```
>>> strng[3:6]
'm l'
>>>
```

Goes to 4 & 5 which are ***space*** & l

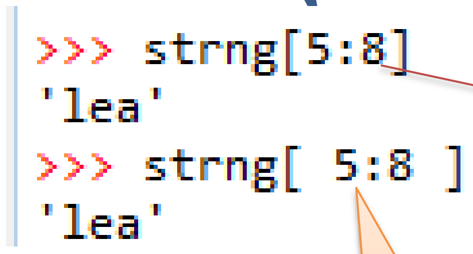
Last number 6, which is 'e' WILL NOT BE INCLUDED

String



```
>>> strng
'I am learning Python'
>>> len(strng)
20
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
...
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
... # 8=r, 9=n, 10=i, 11=n, 12=g, 13=SPACE, 14=P,
... # 15=y, 16=t, 17=h, 18=0, 19=n
```

Start from 5 which is l

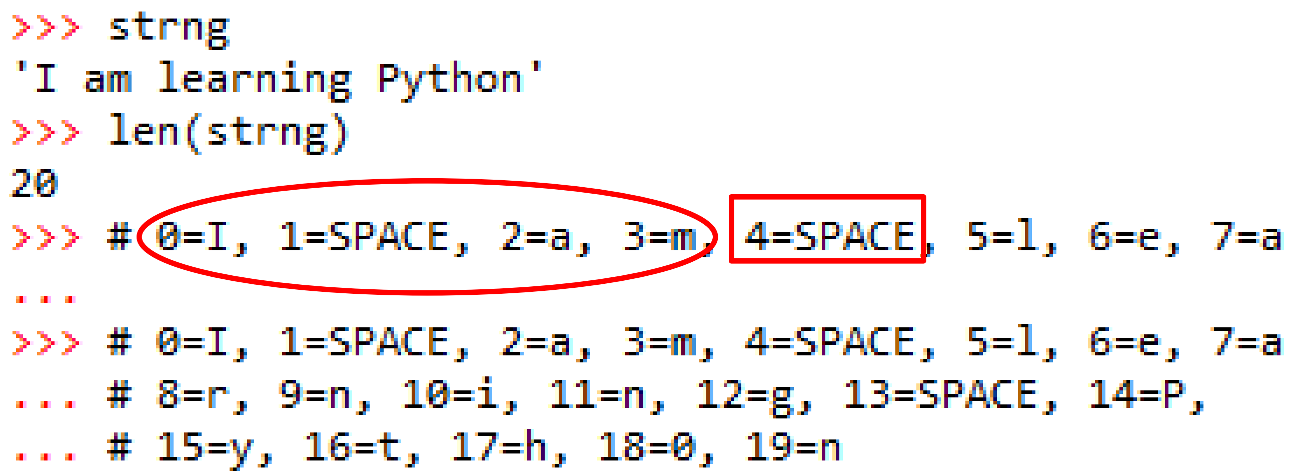


```
>>> strng[5:8]
'lea'
>>> strng[ 5:8 ]
'lea'
```

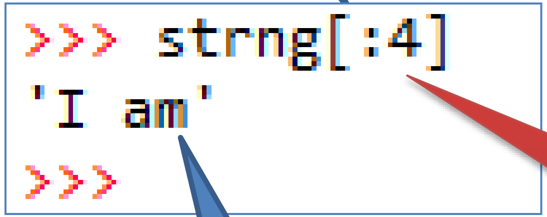
Goes to 6 & 7 which are **e** & **a**

Last number 8, which is 'r' WILL NOT BE INCLUDED

String



```
>>> strng
'I am learning Python'
>>> len(strng)
20
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
...
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
... # 8=r, 9=n, 10=i, 11=n, 12=g, 13=SPACE, 14=P,
... # 15=y, 16=t, 17=h, 18=0, 19=n
```



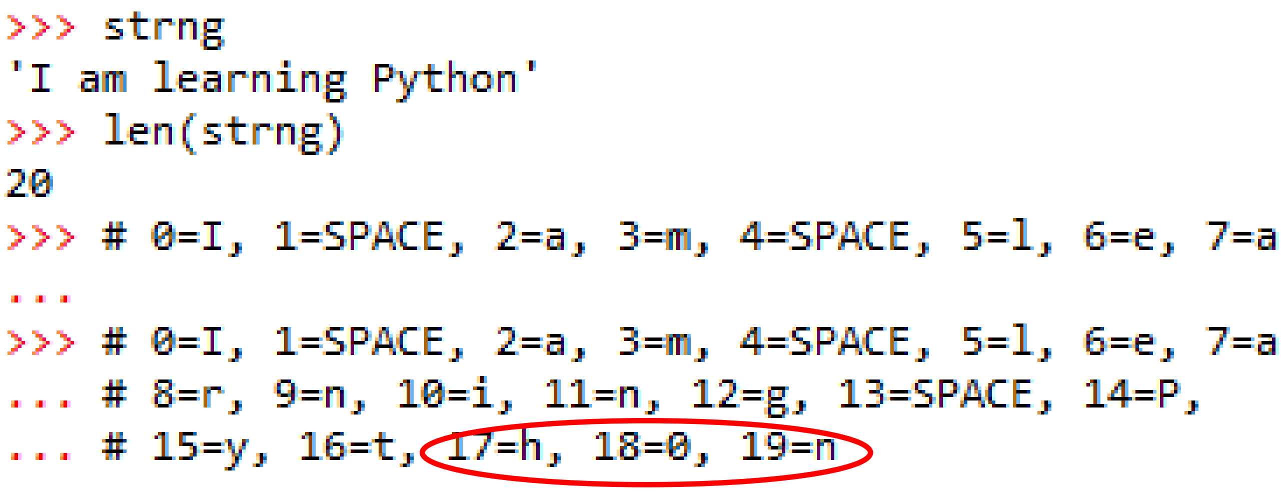
```
>>> strng[:4]
'I am'
>>>
```

Start from 0 which is I

Goes to 3 which is *m*

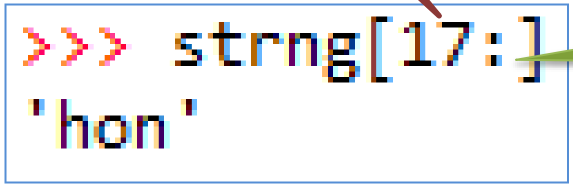
Last number 4, which is *space* WILL NOT BE INCLUDED

String



```
>>> strng
'I am learning Python'
>>> len(strng)
20
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
...
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
... # 8=r, 9=n, 10=i, 11=n, 12=g, 13=SPACE, 14=P,
... # 15=y, 16=t, 17=h, 18=0, 19=n
```

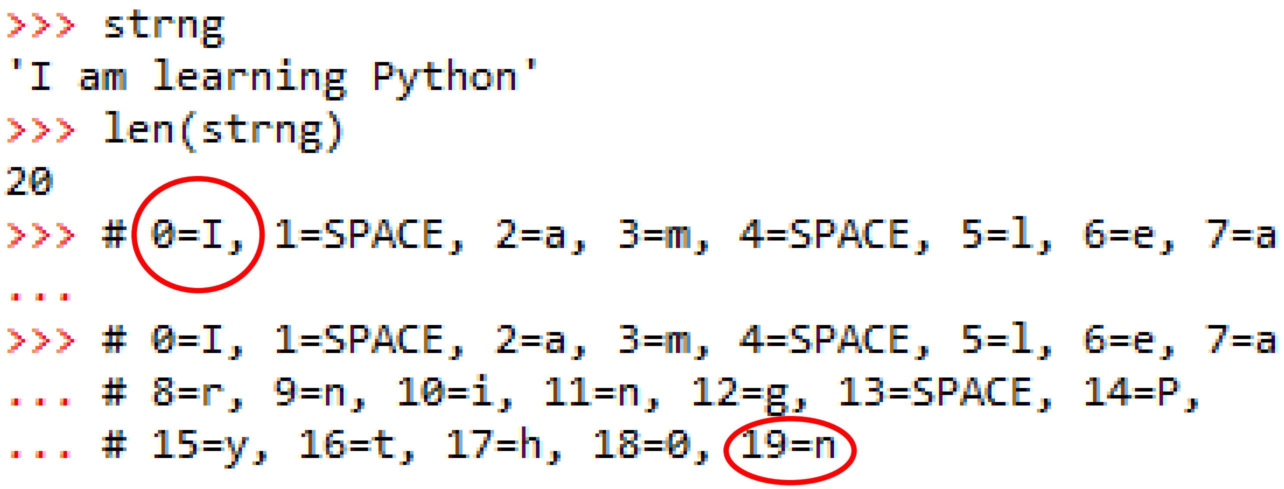
Start from 17 which is h



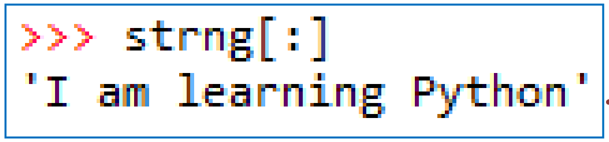
```
>>> strng[17:]
'hon'
```

If after : nothing is mentioned, it will go up to last

String



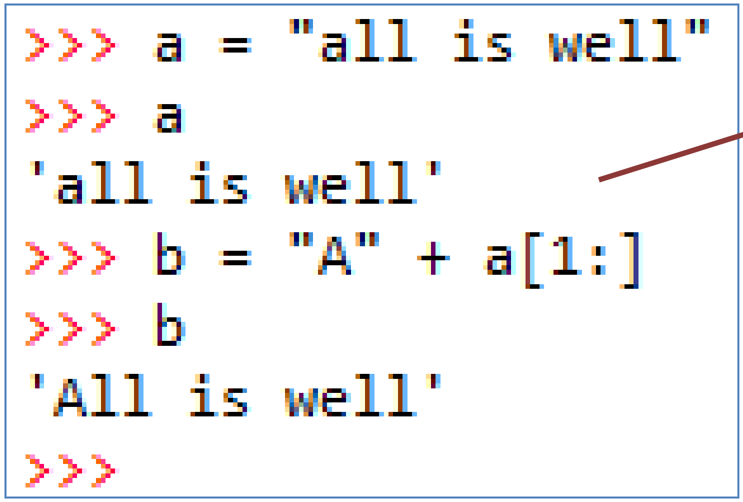
```
>>> strng
'I am learning Python'
>>> len(strng)
20
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
...
>>> # 0=I, 1=SPACE, 2=a, 3=m, 4=SPACE, 5=l, 6=e, 7=a
... # 8=r, 9=n, 10=i, 11=n, 12=g, 13=SPACE, 14=P,
... # 15=y, 16=t, 17=h, 18=0, 19=n
```



```
>>> strng[:]
'I am learning Python'
```

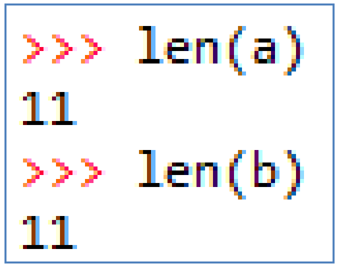
Start from 1st element goes up to last element

Replacing a by A
Adding two strings



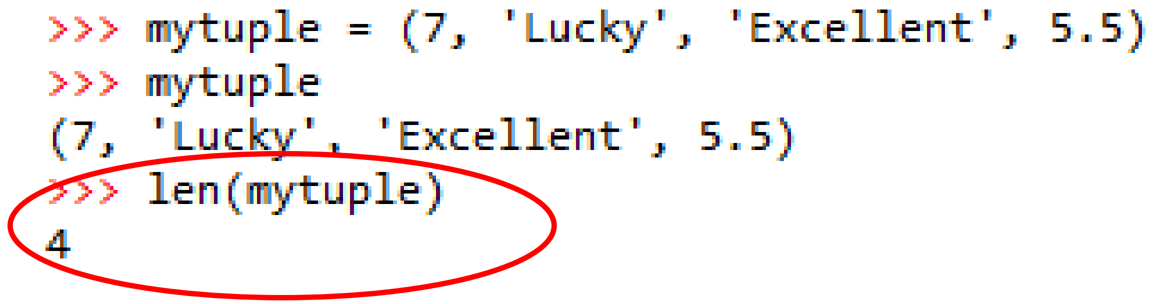
```
>>> a = "all is well"
>>> a
'all is well'
>>> b = "A" + a[1:]
>>> b
'All is well'
>>>
```

A is taking place of a in string a, look how a[1:] is playing role (ignoring a in all)

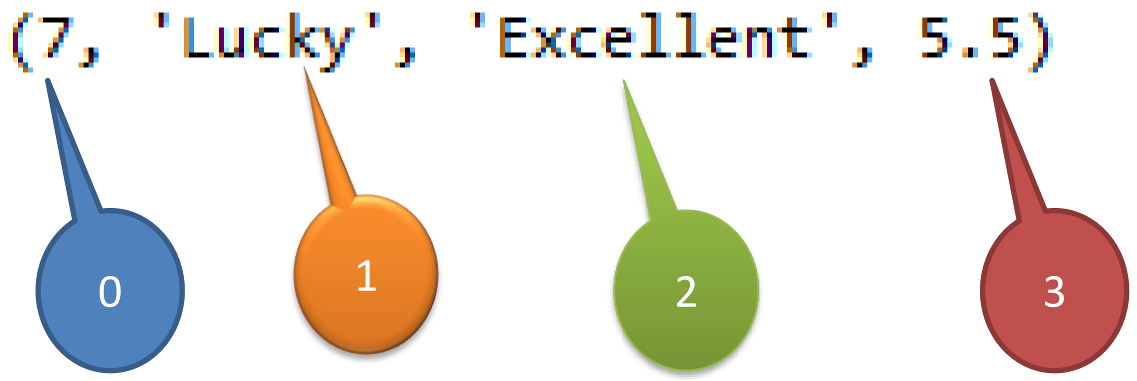


```
>>> len(a)
11
>>> len(b)
11
```

Tuple



```
>>> mytuple = (7, 'Lucky', 'Excellent', 5.5)
>>> mytuple
(7, 'Lucky', 'Excellent', 5.5)
>>> len(mytuple)
4
```

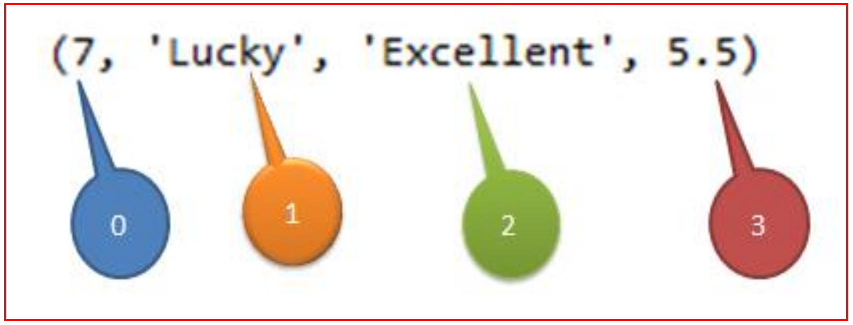


```
(7, 'Lucky', 'Excellent', 5.5)
```

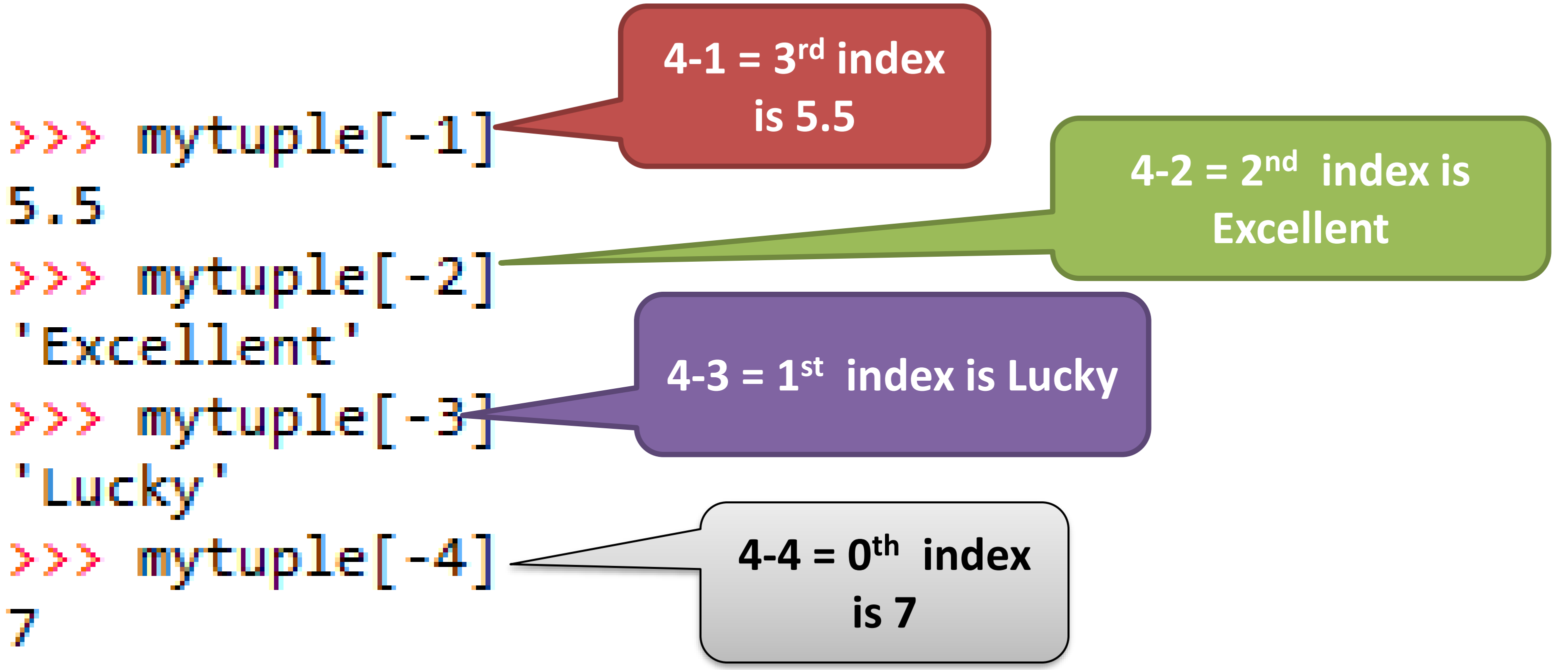
Space between two elements does not matter

Tuple

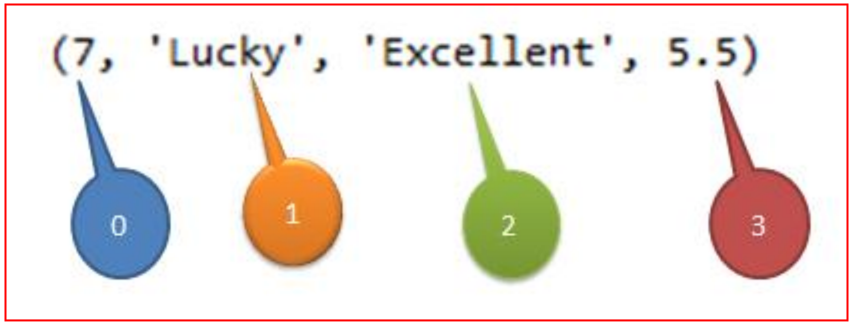
```
>>> mytuple
(7, 'Lucky', 'Excellent', 5.5)
>>> mytuple[0]
7
>>> mytuple[1]
'Lucky'
>>> mytuple[2]
'Excellent'
>>> mytuple[3]
5.5
>>> mytuple[4]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: tuple index out of range
```



Tuple

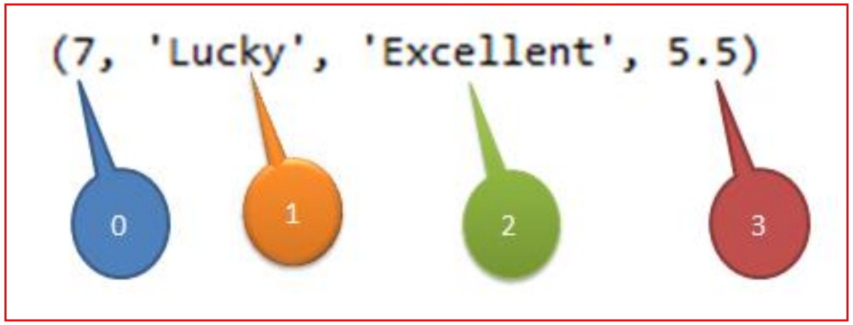


```
>>> mytuple[-1]
5.5
>>> mytuple[-2]
'Excellent'
>>> mytuple[-3]
'Lucky'
>>> mytuple[-4]
7
```

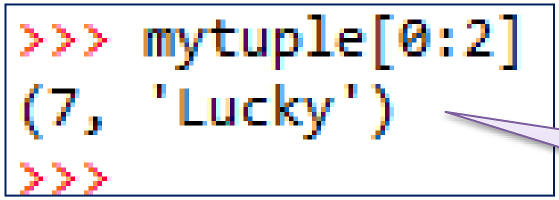


```
(7, 'Lucky', 'Excellent', 5.5)
```

Tuple



```
(7, 'Lucky', 'Excellent', 5.5)
```

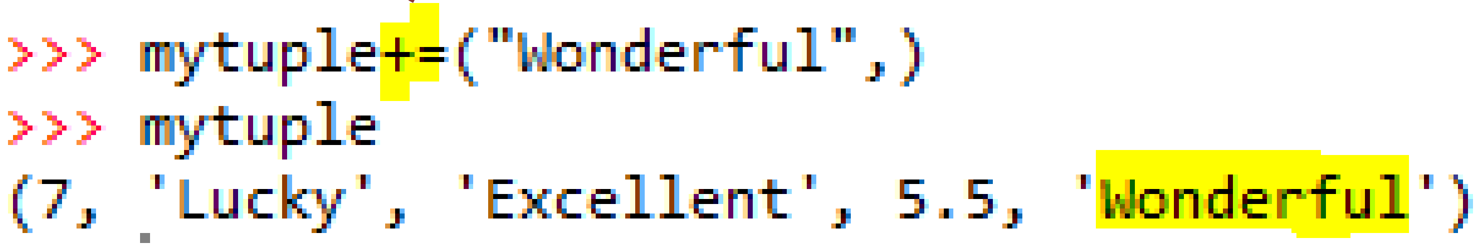


```
>>> mytuple[0:2]
(7, 'Lucky')
>>>
```

0 is 7, 1 is Lucky; **2 which is Excellent is IGNORED**

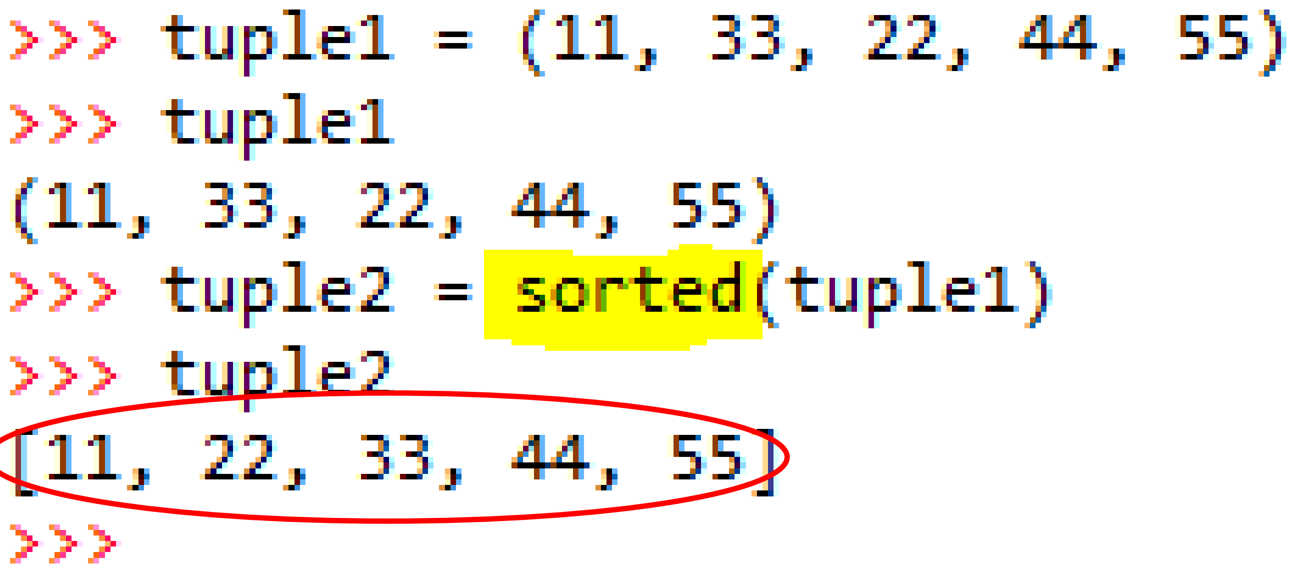
Tuple: adding word ‘Wonderful’

Note the signs
+=



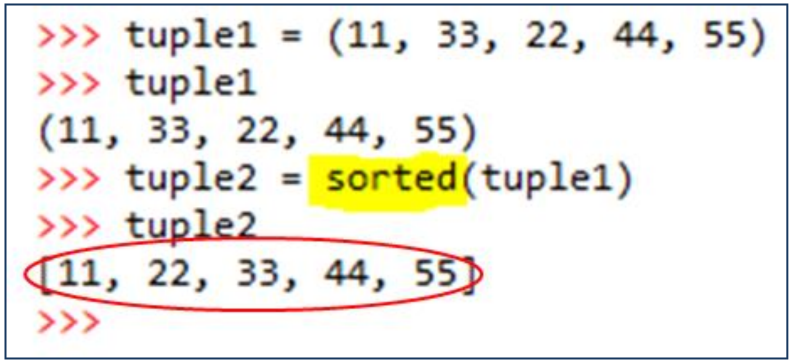
```
>>> mytuple+=("Wonderful",)
>>> mytuple
(7, 'Lucky', 'Excellent', 5.5, 'Wonderful')
```

Tuple: sorting

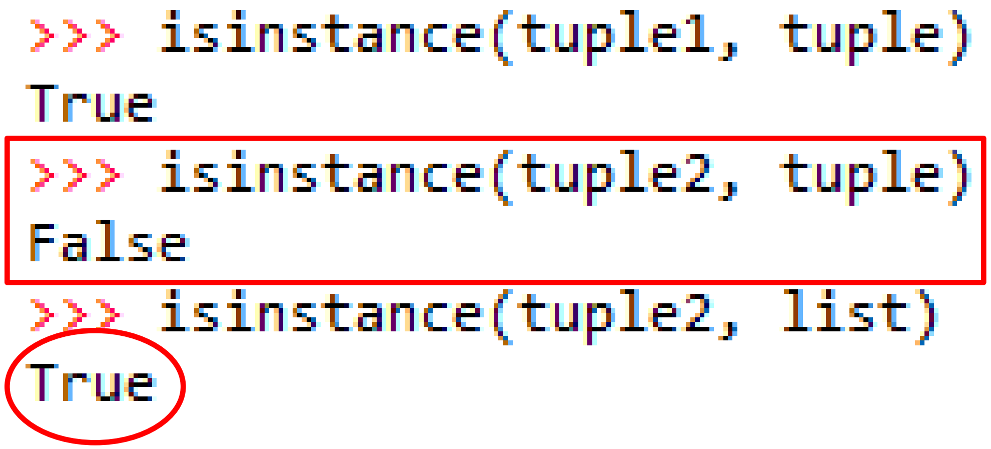


```
>>> tuple1 = (11, 33, 22, 44, 55)
>>> tuple1
(11, 33, 22, 44, 55)
>>> tuple2 = sorted(tuple1)
>>> tuple2
[11, 22, 33, 44, 55]
>>>
```

Sorted tuple is a list

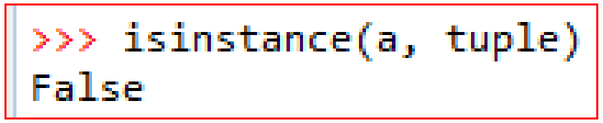


```
>>> tuple1 = (11, 33, 22, 44, 55)
>>> tuple1
(11, 33, 22, 44, 55)
>>> tuple2 = sorted(tuple1)
>>> tuple2
[11, 22, 33, 44, 55]
>>>
```



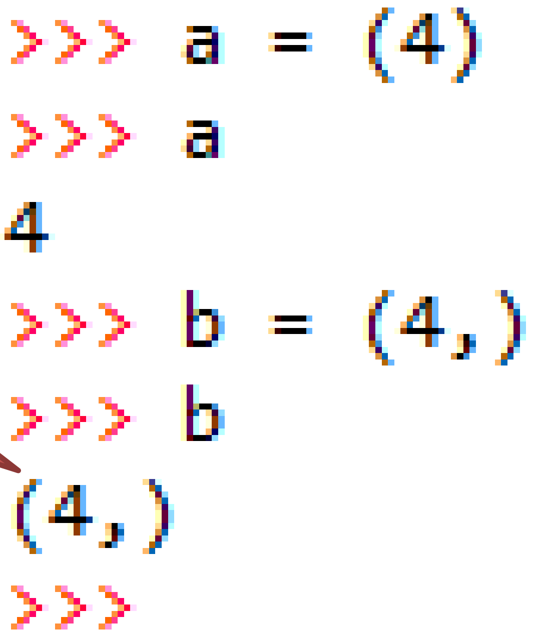
```
>>> isinstance(tuple1, tuple)
True
>>> isinstance(tuple2, tuple)
False
>>> isinstance(tuple2, list)
True
```

Integer vs Tuple



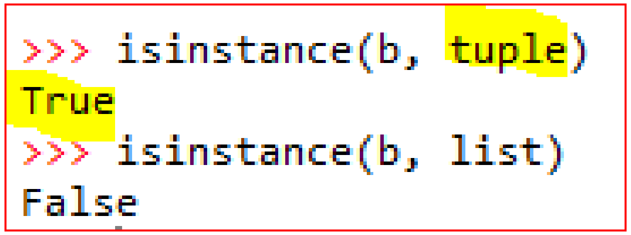
```
>>> isinstance(a, tuple)
False
```

a is holding an integer 4



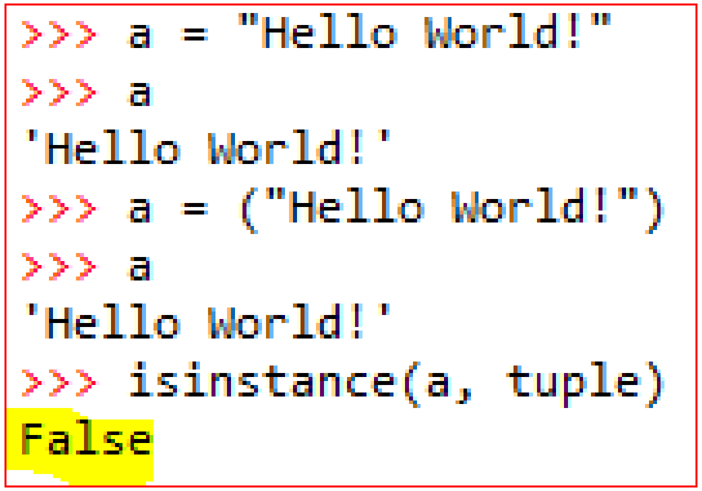
```
>>> a = (4)
>>> a
4
>>> b = (4,)
>>> b
(4,)
>>>
```

b is a tuple having 4 in it

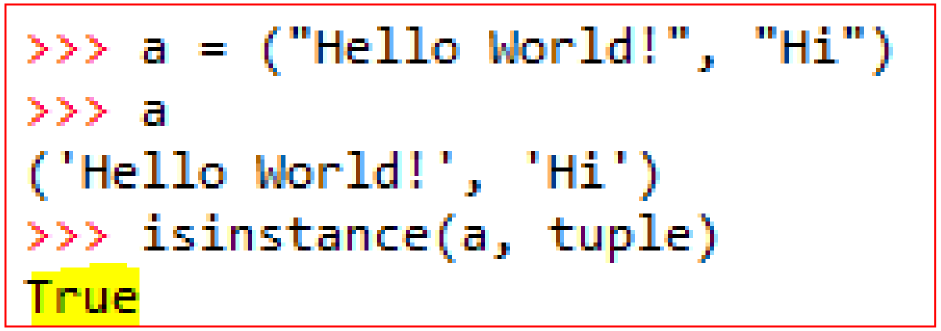


```
>>> isinstance(b, tuple)
True
>>> isinstance(b, list)
False
```

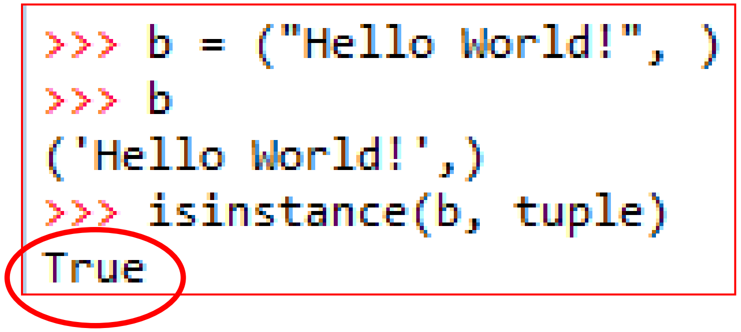
String vs tuple



```
>>> a = "Hello World!"
>>> a
'Hello World!'
>>> a = ("Hello World!")
>>> a
'Hello World!'
>>> isinstance(a, tuple)
False
```

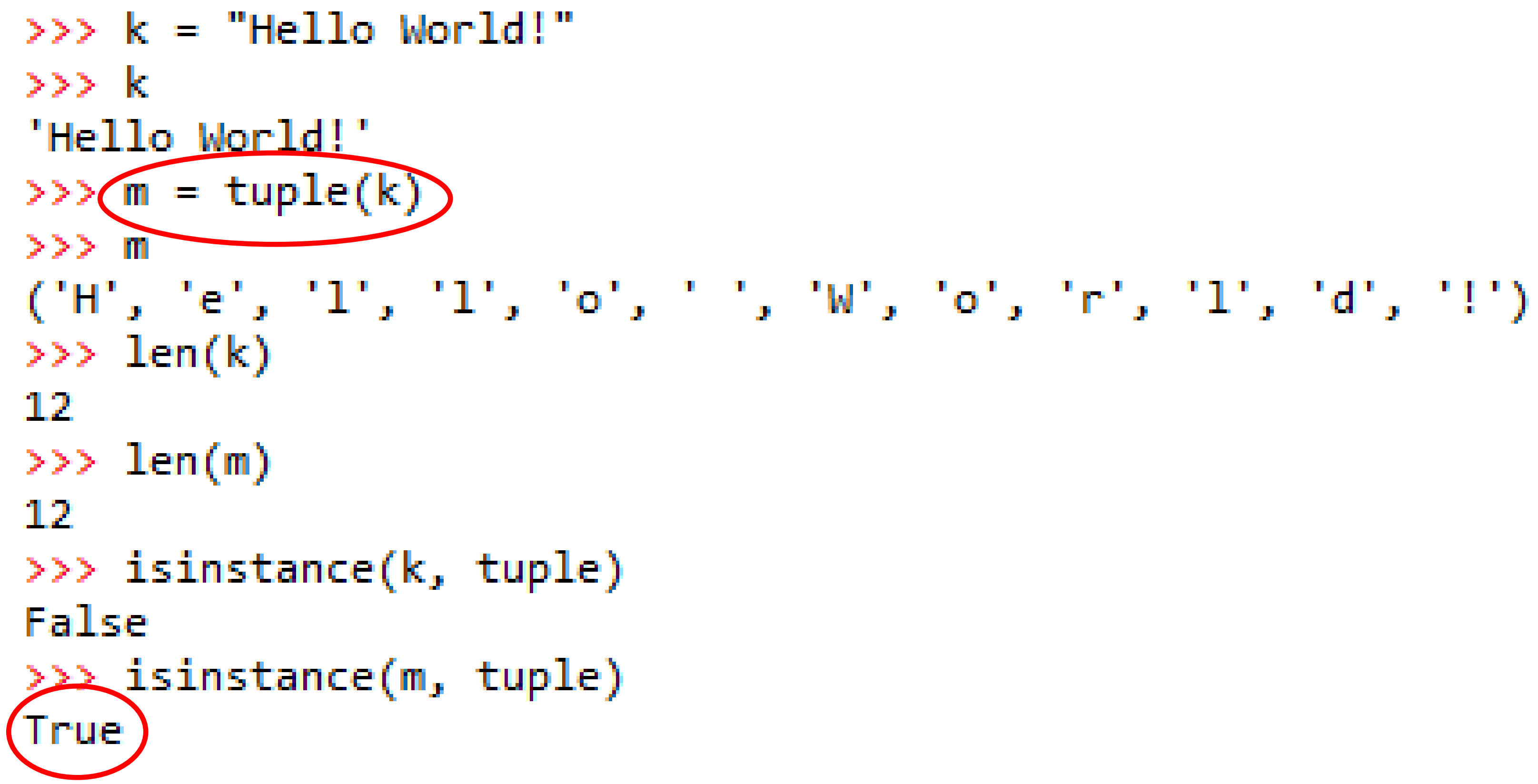


```
>>> a = ("Hello World!", "Hi")
>>> a
('Hello World!', 'Hi')
>>> isinstance(a, tuple)
True
```



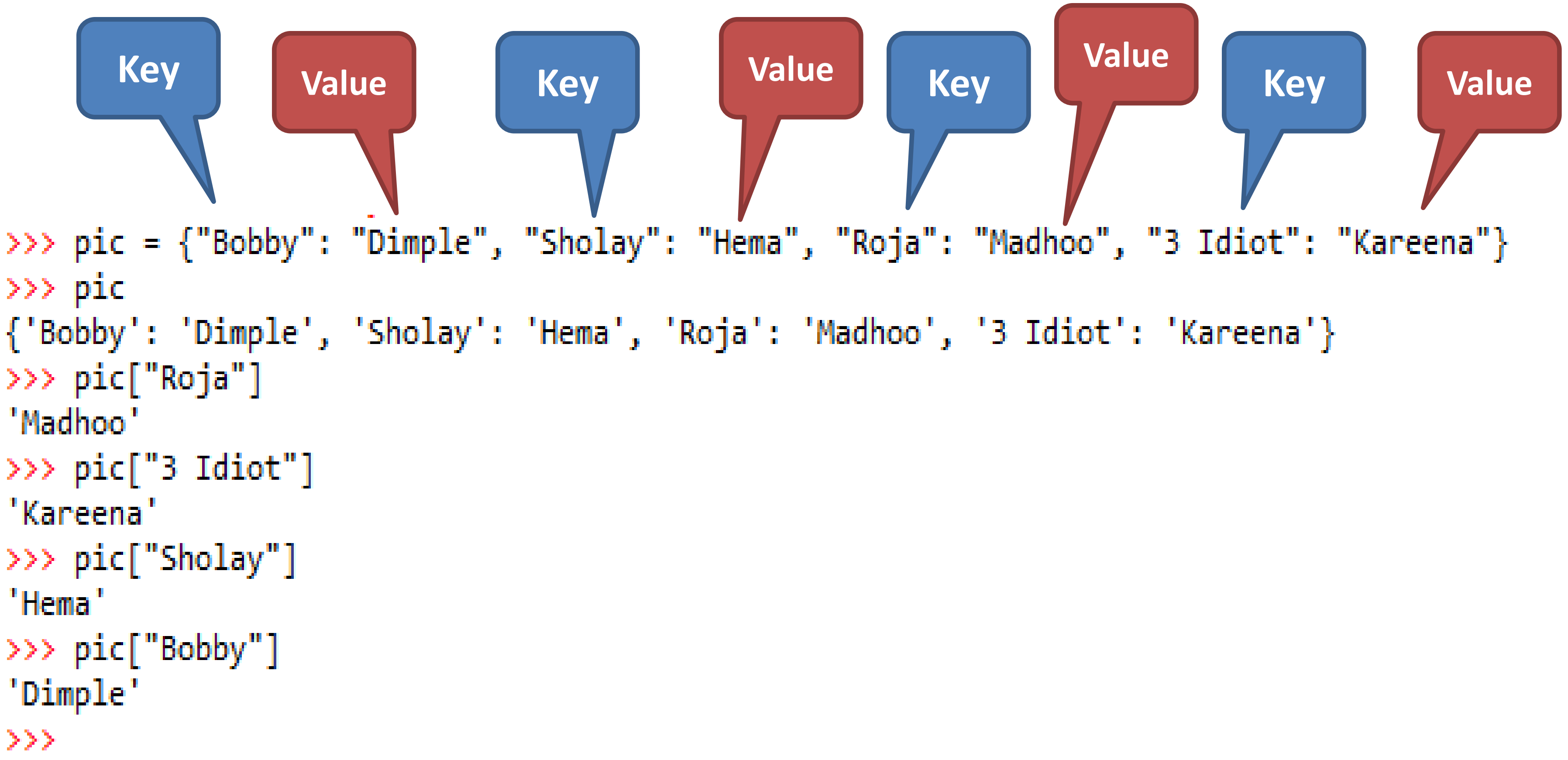
```
>>> b = ("Hello World!", )
>>> b
('Hello World!',)
>>> isinstance(b, tuple)
True
```

String into tuple



```
>>> k = "Hello World!"
>>> k
'Hello World!'
>>> m = tuple(k)
>>> m
('H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!')
>>> len(k)
12
>>> len(m)
12
>>> isinstance(k, tuple)
False
>>> isinstance(m, tuple)
True
```

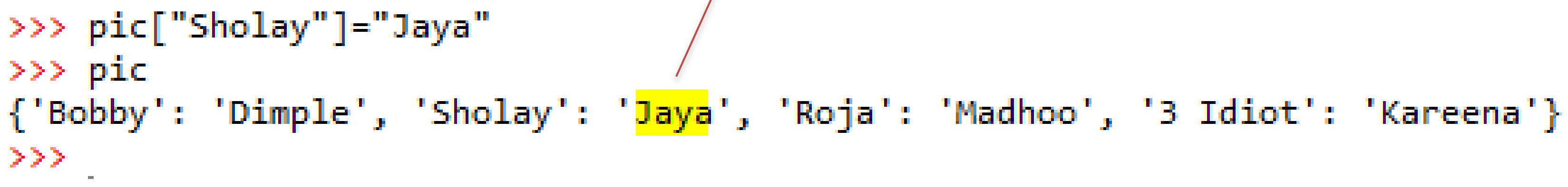
Dictionary



```
>>> pic = {"Bobby": "Dimple", "Sholay": "Hema", "Roja": "Madhoo", "3 Idiot": "Kareena"}
>>> pic
{'Bobby': 'Dimple', 'Sholay': 'Hema', 'Roja': 'Madhoo', '3 Idiot': 'Kareena'}
>>> pic["Roja"]
'Madhoo'
>>> pic["3 Idiot"]
'Kareena'
>>> pic["Sholay"]
'Hema'
>>> pic["Bobby"]
'Dimple'
>>>
```

You can change Value to a key

Hema replaced by Jaya



```
>>> pic["Sholay"]="Jaya"
>>> pic
{'Bobby': 'Dimple', 'Sholay': 'Jaya', 'Roja': 'Madhoo', '3 Idiot': 'Kareena'}
>>>
```

Create new pair

```
>>> pic["Dangal"] = "Sana"
```

New Keys

New Values

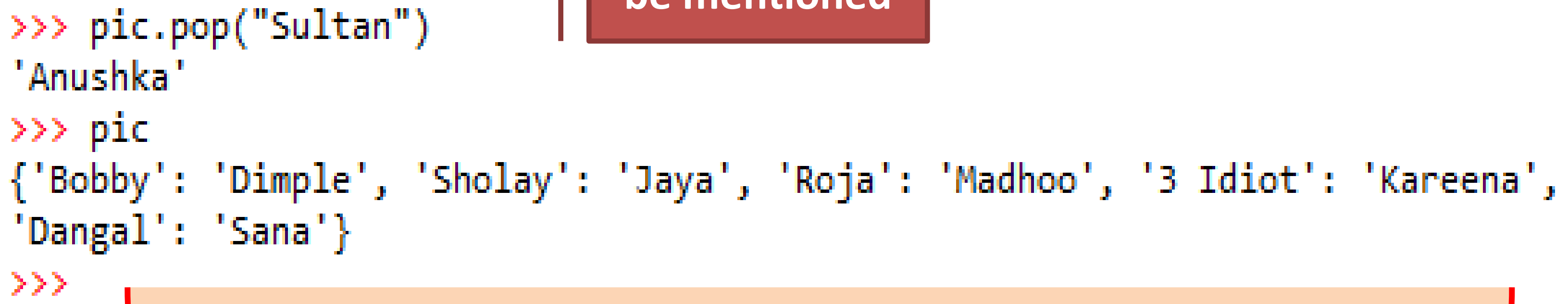
```
>>> pic["Sultan"] = "Anushka"
```

```
>>> pic
{'Bobby': 'Dimple', 'Sholay': 'Jaya', 'Roja': 'Madhoo', '3 Idiot': 'Kareena',
'Dangal': 'Sana', 'Sultan': 'Anushka'}
>>>
```

New Dictionary

Remove any item say, Sultan:Anhushka

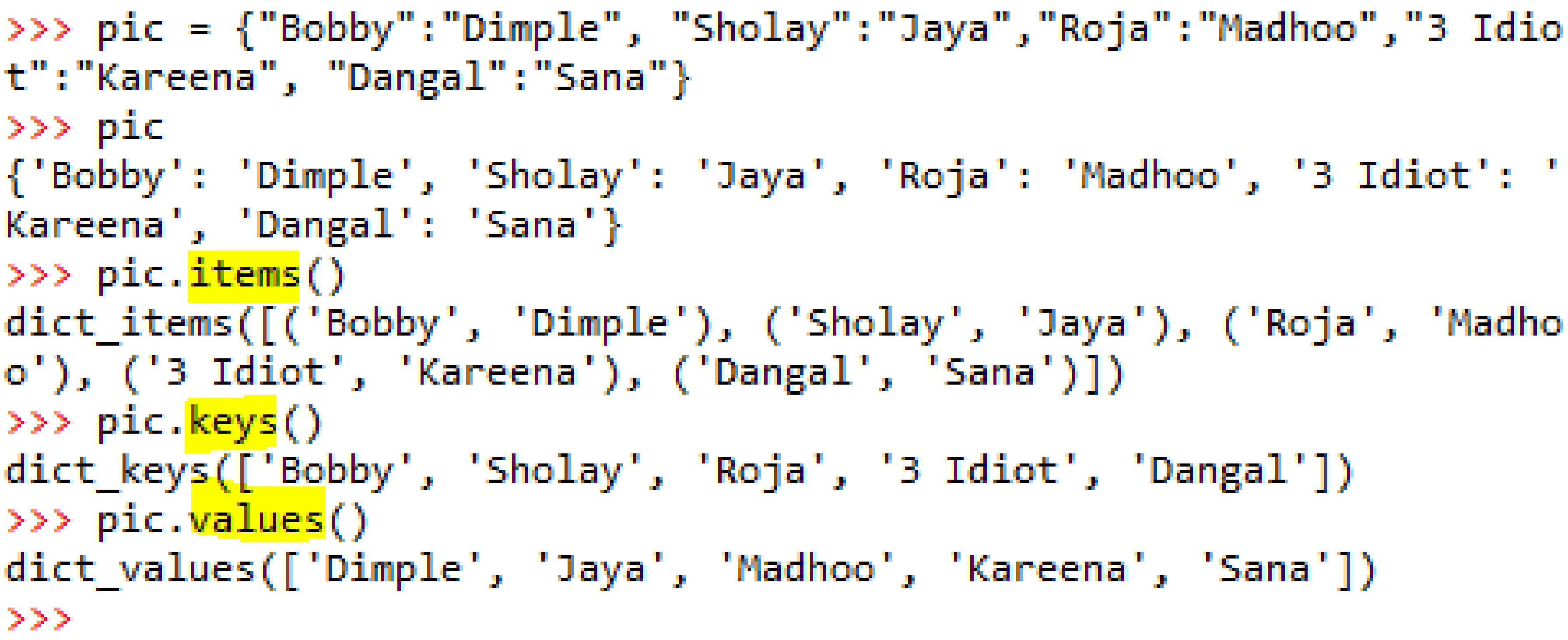
Use command <name>.pop ONLY key is to be mentioned



```
>>> pic.pop("Sultan")
'Anushka'
>>> pic
{'Bobby': 'Dimple', 'Sholay': 'Jaya', 'Roja': 'Madhoo', '3 Idiot': 'Kareena',
'Dangal': 'Sana'}
>>>
```

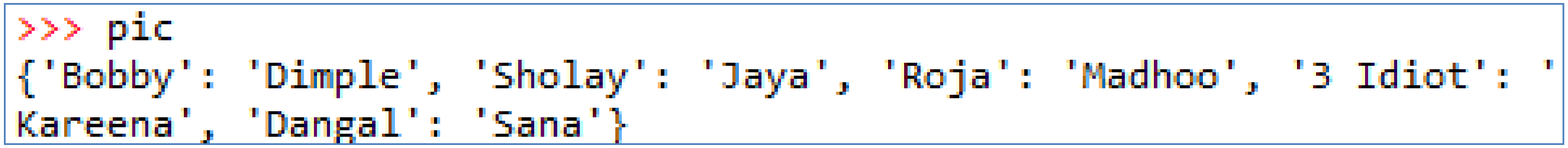
New Dictionary

Retrieve Items, Keys, Values

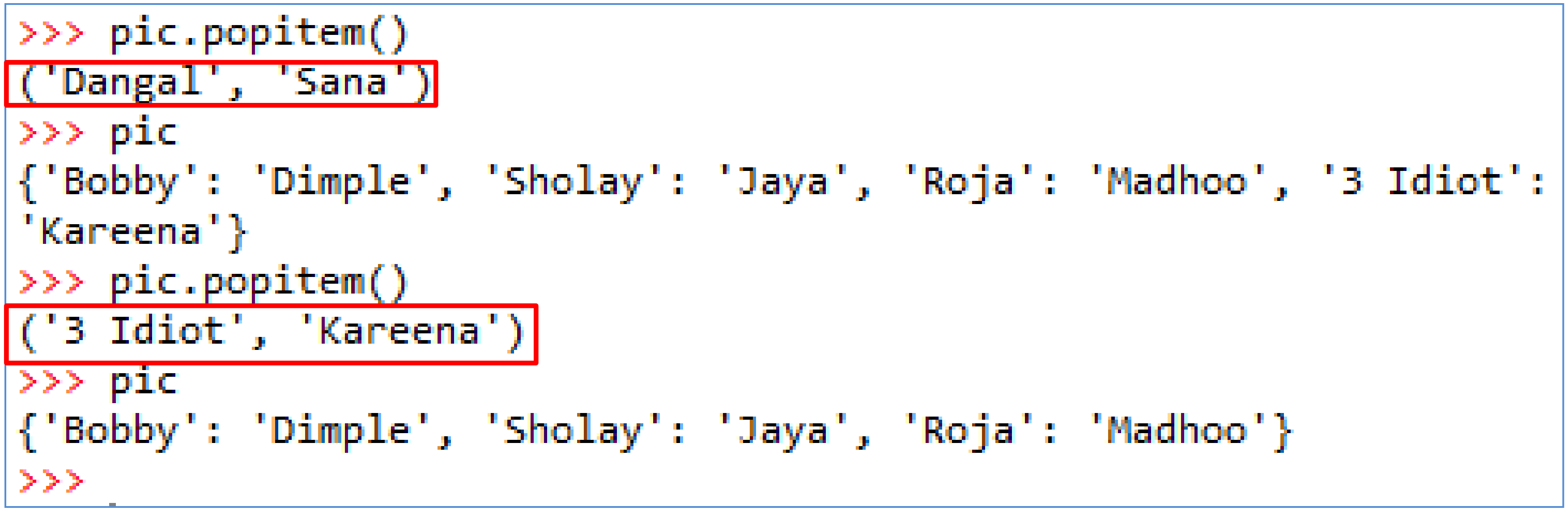


```
>>> pic = {"Bobby":"Dimple", "Sholay":"Jaya","Roja":"Madhoo","3 Idio
t":"Kareena", "Dangal":"Sana"}
>>> pic
{'Bobby': 'Dimple', 'Sholay': 'Jaya', 'Roja': 'Madhoo', '3 Idiot': '
Kareena', 'Dangal': 'Sana'}
>>> pic.items()
dict_items([('Bobby', 'Dimple'), ('Sholay', 'Jaya'), ('Roja', 'Madho
o'), ('3 Idiot', 'Kareena'), ('Dangal', 'Sana')])
>>> pic.keys()
dict_keys(['Bobby', 'Sholay', 'Roja', '3 Idiot', 'Dangal'])
>>> pic.values()
dict_values(['Dimple', 'Jaya', 'Madhoo', 'Kareena', 'Sana'])
>>>
```

Removing LAST items

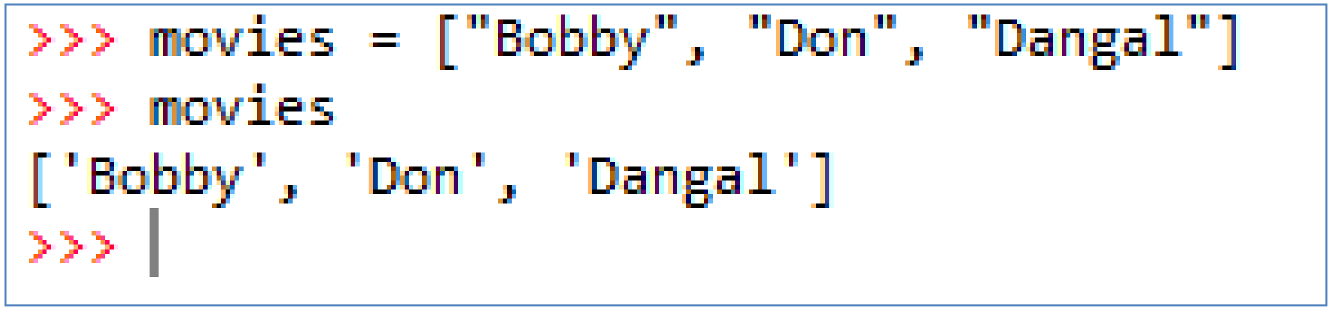


```
>>> pic
{'Bobby': 'Dimple', 'Sholay': 'Jaya', 'Roja': 'Madhoo', '3 Idiot': '
Kareena', 'Dangal': 'Sana'}
```



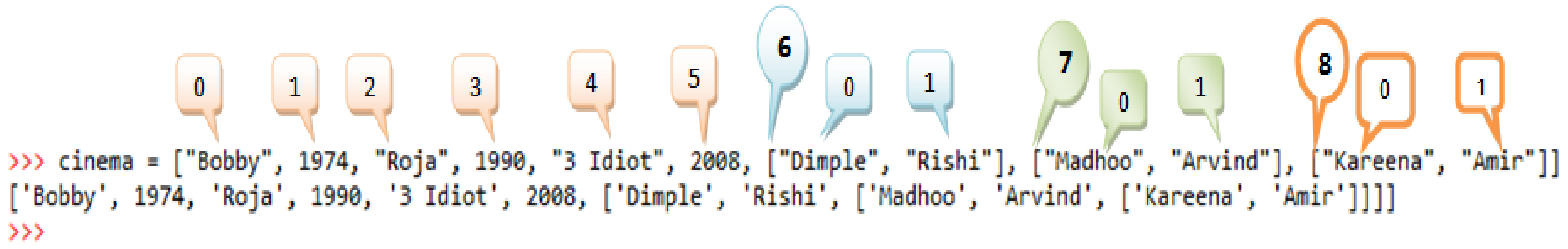
```
>>> pic.popitem()
('Dangal', 'Sana')
>>> pic
{'Bobby': 'Dimple', 'Sholay': 'Jaya', 'Roja': 'Madhoo', '3 Idiot':
'Kareena'}
>>> pic.popitem()
('3 Idiot', 'Kareena')
>>> pic
{'Bobby': 'Dimple', 'Sholay': 'Jaya', 'Roja': 'Madhoo'}
>>>
```

List of movies



```
>>> movies = ["Bobby", "Don", "Dangal"]
>>> movies
['Bobby', 'Don', 'Dangal']
>>> 
```

Lists

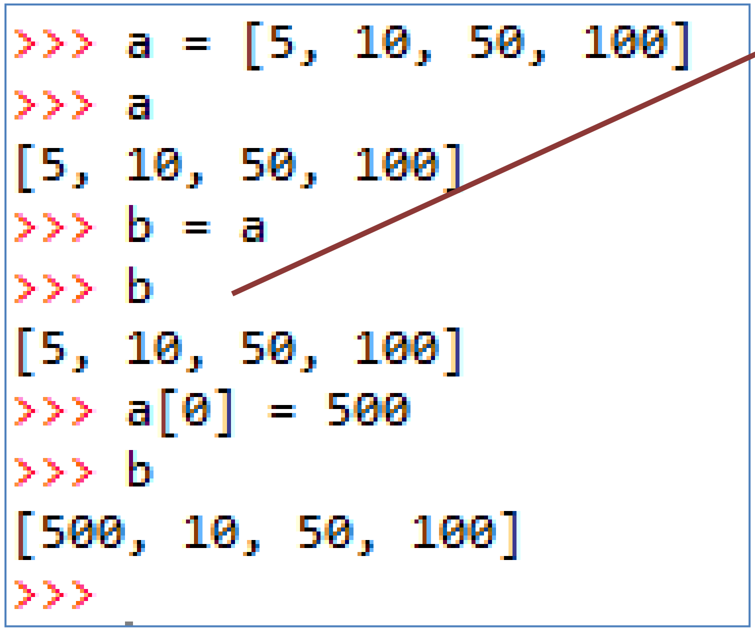


```
>>> cinema = ["Bobby", 1974, "Roja", 1990, "3 Idiot", 2008, ["Dimple", "Rishi"], ["Madhoo", "Arvind"], ["Kareena", "Amir"]]
['Bobby', 1974, 'Roja', 1990, '3 Idiot', 2008, ['Dimple', 'Rishi'], ['Madhoo', 'Arvind'], ['Kareena', 'Amir']]]
>>>
```

`>>> print(cinema[2])`
`>>>`
`>>>`
`Roja`
	`>>> print(cinema[4])`
	`>>>`
	`>>>`
	`3 Idiot`
		`>>> print(cinema[7][0])`
		`>>>`
		`>>>`
		`Madhoo`
			`>>> print(cinema[8][0])`
			`>>>`
			`>>>`
			`Kareena`

`>>> print(cinema[1])`
`>>>`
`>>>`
`1974`
	`>>> print(cinema[6][0])`
	`>>>`
	`>>>`
	`Dimple`
		`>>> print(cinema[7][1])`
		`>>>`
		`>>>`
		`Arvind`
			`>>> print(cinema[8][1])`
			`>>>`
			`>>>`
			`Amir`

Alias



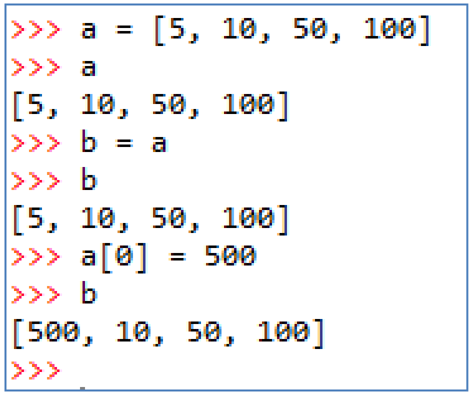
```
>>> a = [5, 10, 50, 100]
>>> a
[5, 10, 50, 100]
>>> b = a
>>> b
[5, 10, 50, 100]
>>> a[0] = 500
>>> b
[500, 10, 50, 100]
>>>
```

b is alias of a
Change in a is
reflected in b

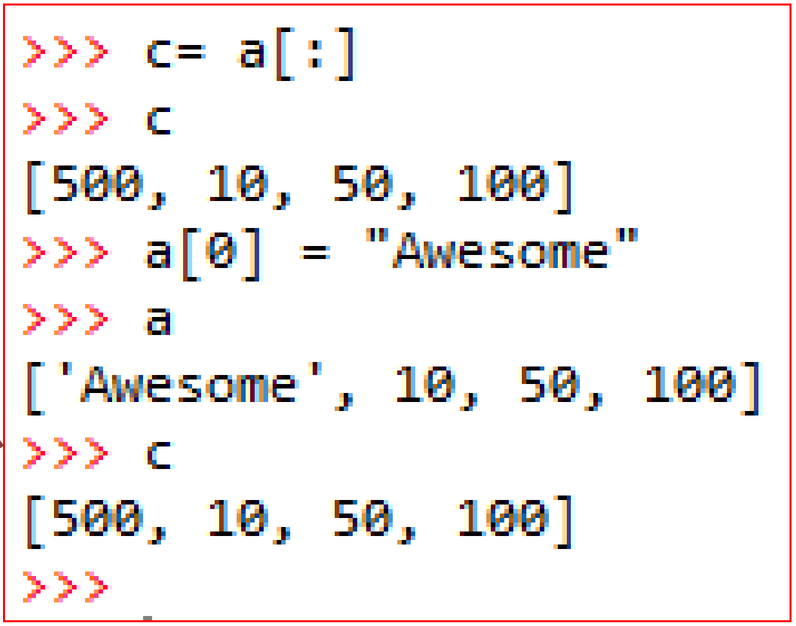
Clone

c is a clone
Change in a is not resulting change in c

b is alias of a

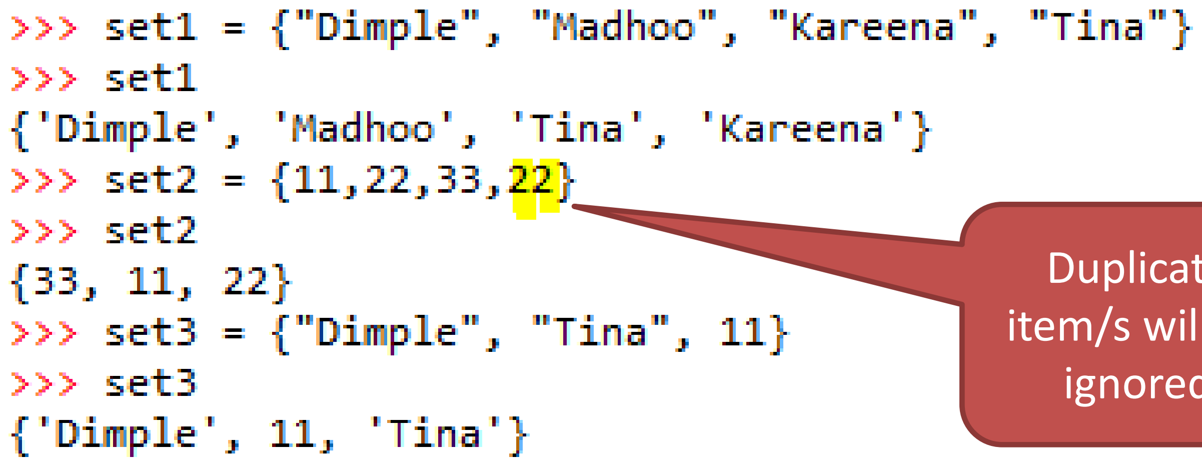


```
>>> a = [5, 10, 50, 100]
>>> a
[5, 10, 50, 100]
>>> b = a
>>> b
[5, 10, 50, 100]
>>> a[0] = 500
>>> b
[500, 10, 50, 100]
>>>
```



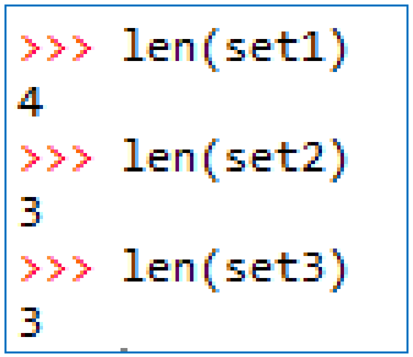
```
>>> c= a[:]
>>> c
[500, 10, 50, 100]
>>> a[0] = "Awesome"
>>> a
['Awesome', 10, 50, 100]
>>> c
[500, 10, 50, 100]
>>>
```

Sets



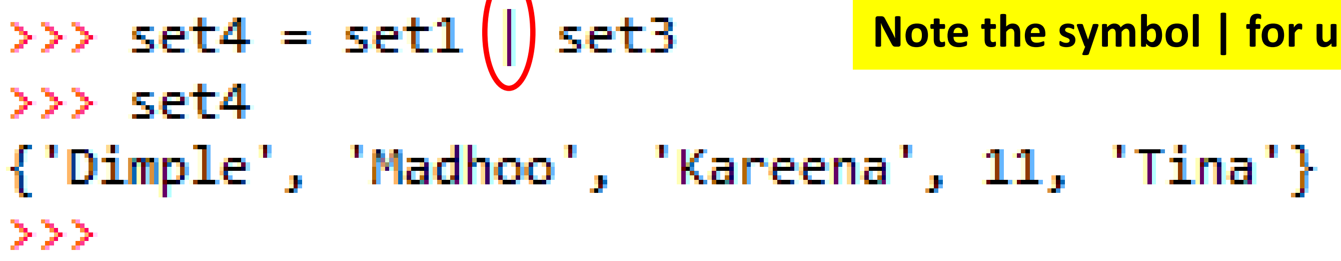
```
>>> set1 = {"Dimple", "Madhoo", "Kareena", "Tina"}
>>> set1
{'Dimple', 'Madhoo', 'Tina', 'Kareena'}
>>> set2 = {11,22,33,22}
>>> set2
{33, 11, 22}
>>> set3 = {"Dimple", "Tina", 11}
>>> set3
{'Dimple', 11, 'Tina'}
```

Duplicate item/s will be ignored



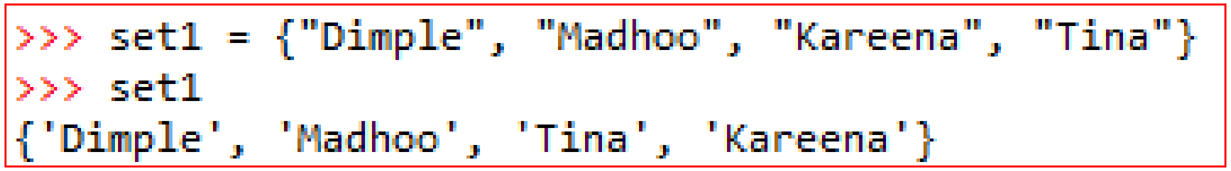
```
>>> len(set1)
4
>>> len(set2)
3
>>> len(set3)
3
```

Union of 2 sets



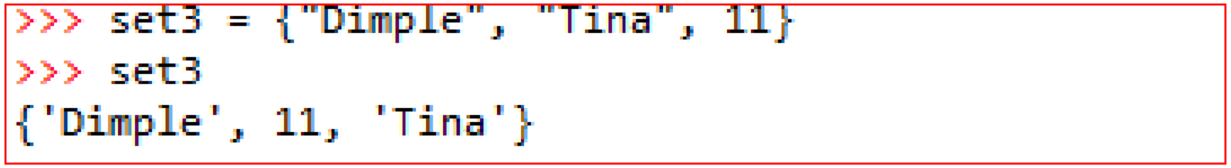
```
>>> set4 = set1 | set3
>>> set4
{'Dimple', 'Madhoo', 'Kareena', 11, 'Tina'}
>>>
```

Note the symbol | for union



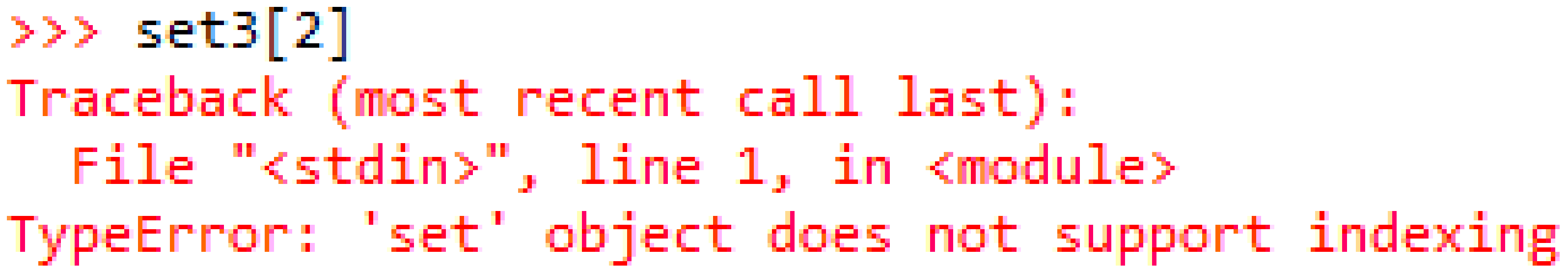
```
>>> set1 = {"Dimple", "Madhoo", "Kareena", "Tina"}
>>> set1
{'Dimple', 'Madhoo', 'Tina', 'Kareena'}
```

Duplicates are ignored

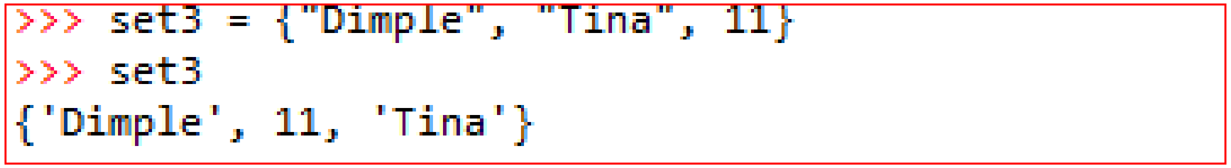


```
>>> set3 = {"Dimple", "Tina", 11}
>>> set3
{'Dimple', 11, 'Tina'}
```

Sets do not have orders

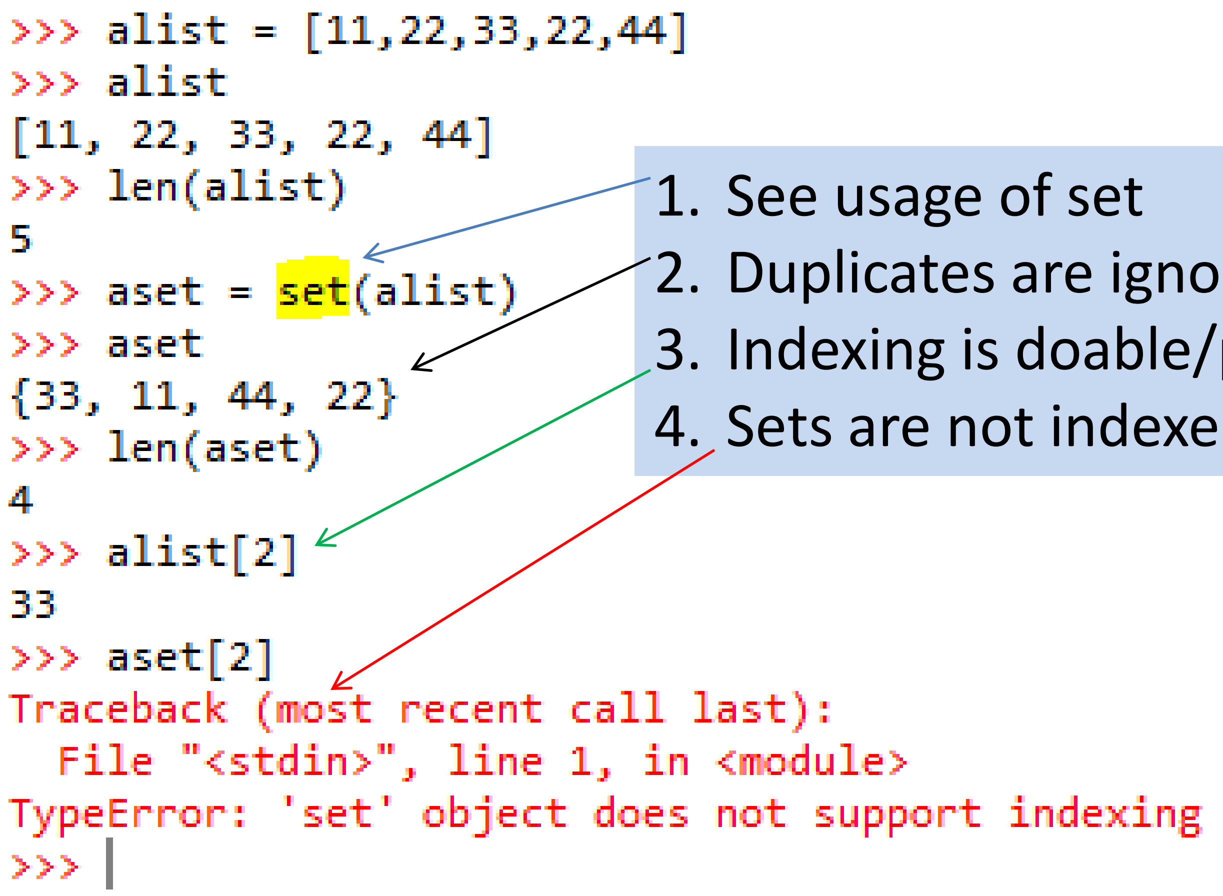


```
>>> set3[2]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object does not support indexing
```



```
>>> set3 = {"Dimple", "Tina", 11}
>>> set3
{'Dimple', 11, 'Tina'}
```

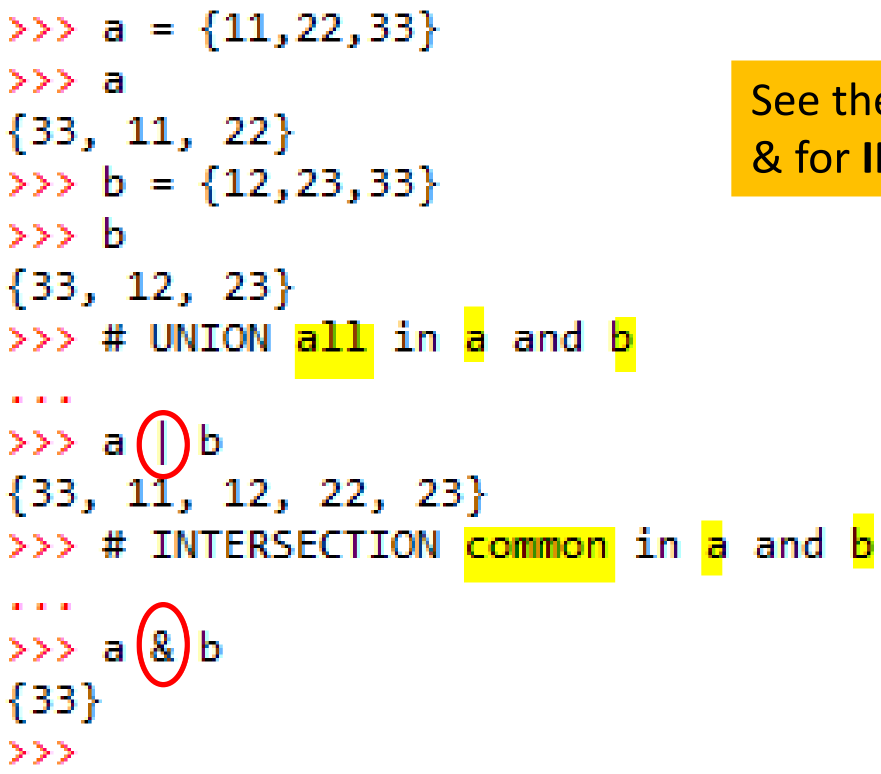
List into a Set



```
>>> alist = [11,22,33,22,44]
>>> alist
[11, 22, 33, 22, 44]
>>> len(alist)
5
>>> aset = set(alist)
>>> aset
{33, 11, 44, 22}
>>> len(aset)
4
>>> alist[2]
33
>>> aset[2]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object does not support indexing
>>>
```

1. See usage of set
2. Duplicates are ignored in sets
3. Indexing is doable/possible in LIST
4. Sets are not indexed

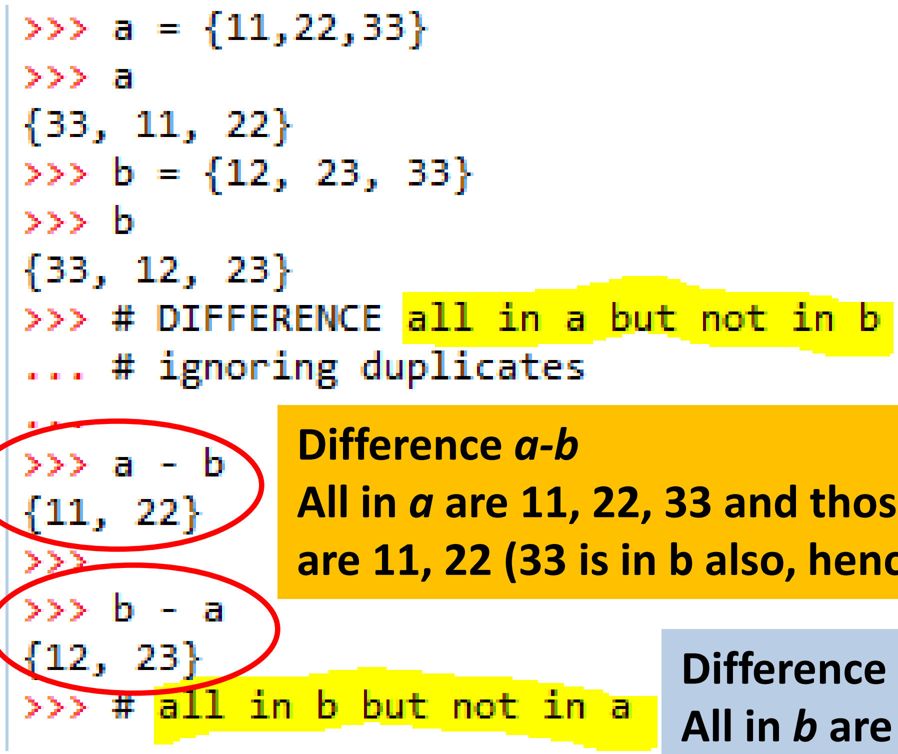
Union and Intersection of sets



```
>>> a = {11,22,33}
>>> a
{33, 11, 22}
>>> b = {12,23,33}
>>> b
{33, 12, 23}
>>> # UNION all in a and b
...
>>> a | b
{33, 11, 12, 22, 23}
>>> # INTERSECTION common in a and b
...
>>> a & b
{33}
>>>
```

See the symbols | for **UNION** and & for **INTERSETION**

Difference



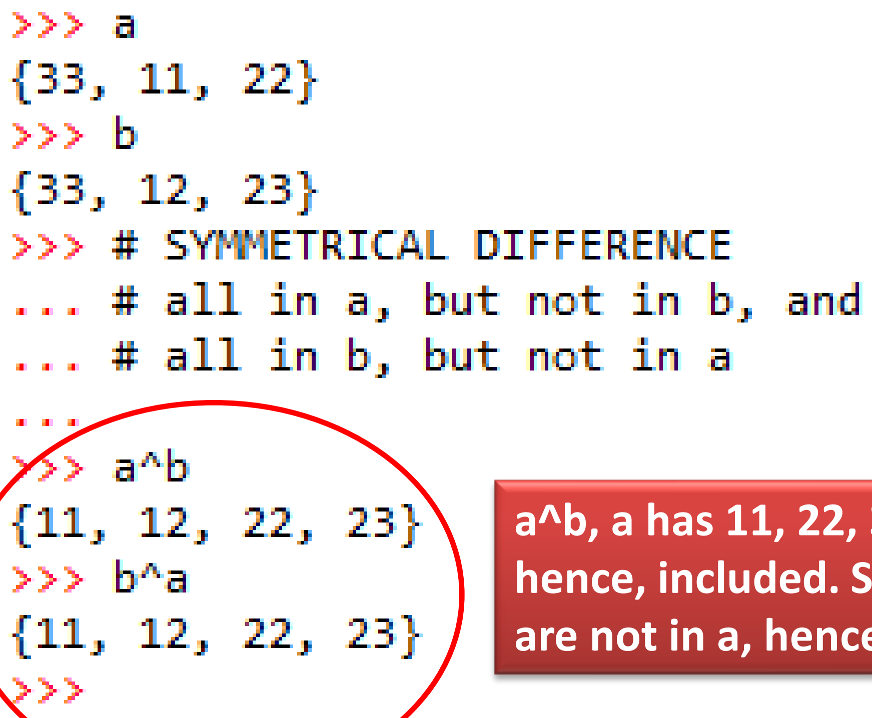
```
>>> a = {11,22,33}
>>> a
{33, 11, 22}
>>> b = {12, 23, 33}
>>> b
{33, 12, 23}
>>> # DIFFERENCE all in a but not in b
... # ignoring duplicates
...
>>> a - b
{11, 22}
>>>
>>> b - a
{12, 23}
>>> # all in b but not in a
```

Bit tricky!

Difference *a-b*
All in *a* are 11, 22, 33 and those are not in *b* are 11, 22 (33 is in b also, hence, ignored)

Difference *b-a*
All in *b* are 12, 23, 33 and those are not in *a* are 12, 23 (33 is in a also, hence, ignored)

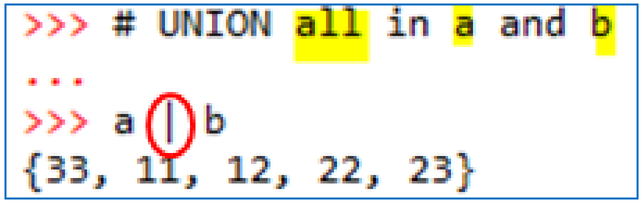
Symmetrical Difference



```
>>> a
{33, 11, 22}
>>> b
{33, 12, 23}
>>> # SYMMETRICAL DIFFERENCE
... # all in a, but not in b, and
... # all in b, but not in a
...
>>> a^b
{11, 12, 22, 23}
>>> b^a
{11, 12, 22, 23}
>>>
```

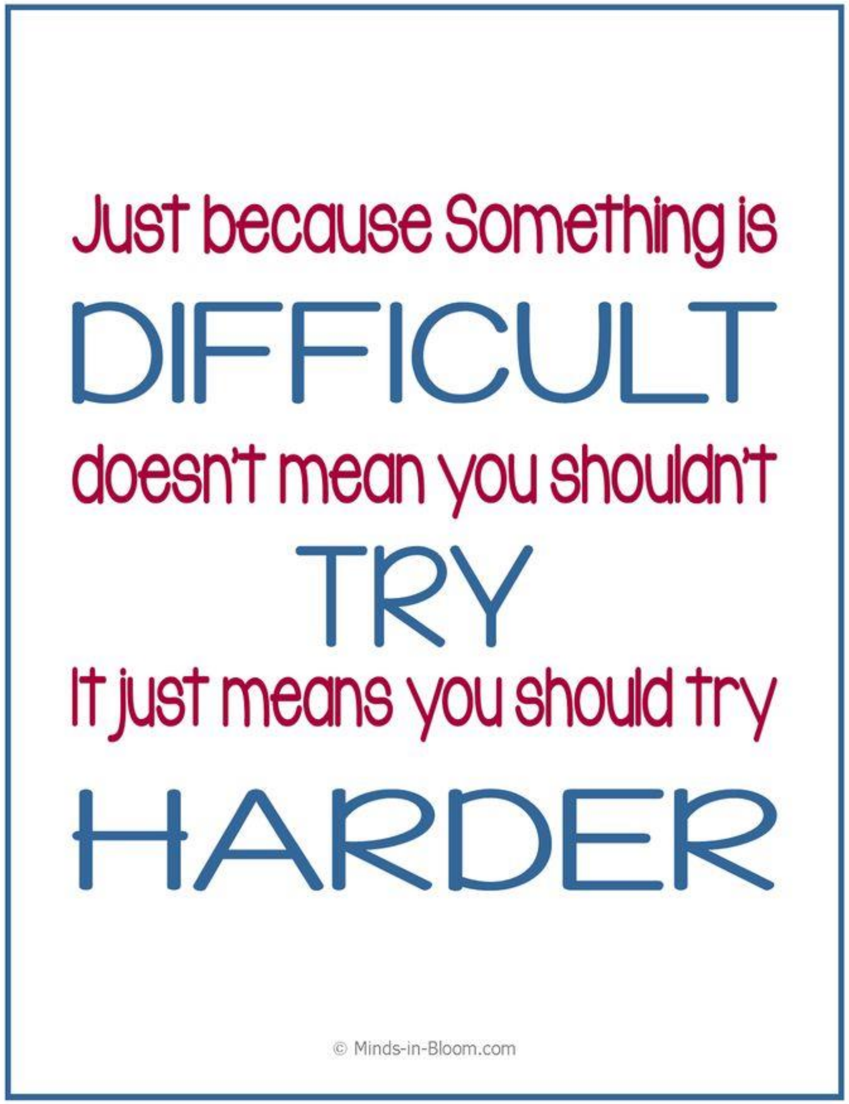
Indeed tricky!

a^b, a has 11, 22, 33 and 11 and 22 are not in b, hence, included. Similarly, b has 12, 23, 33 and 12 and 22 are not in a, hence, included. Poor 33 is ignored!



```
>>> # UNION all in a and b
...
>>> a | b
{33, 11, 12, 22, 23}
```

Happy
Learning


Just because Something is

DIFFICULT

doesn't mean you shouldn't

TRY

It just means you should try

HARDER